PROJECTS ■ THEORY ■ APPLICATIONS ■ CIRCUITS ■ TECHNOLOGY

NUTS AND VOLTS

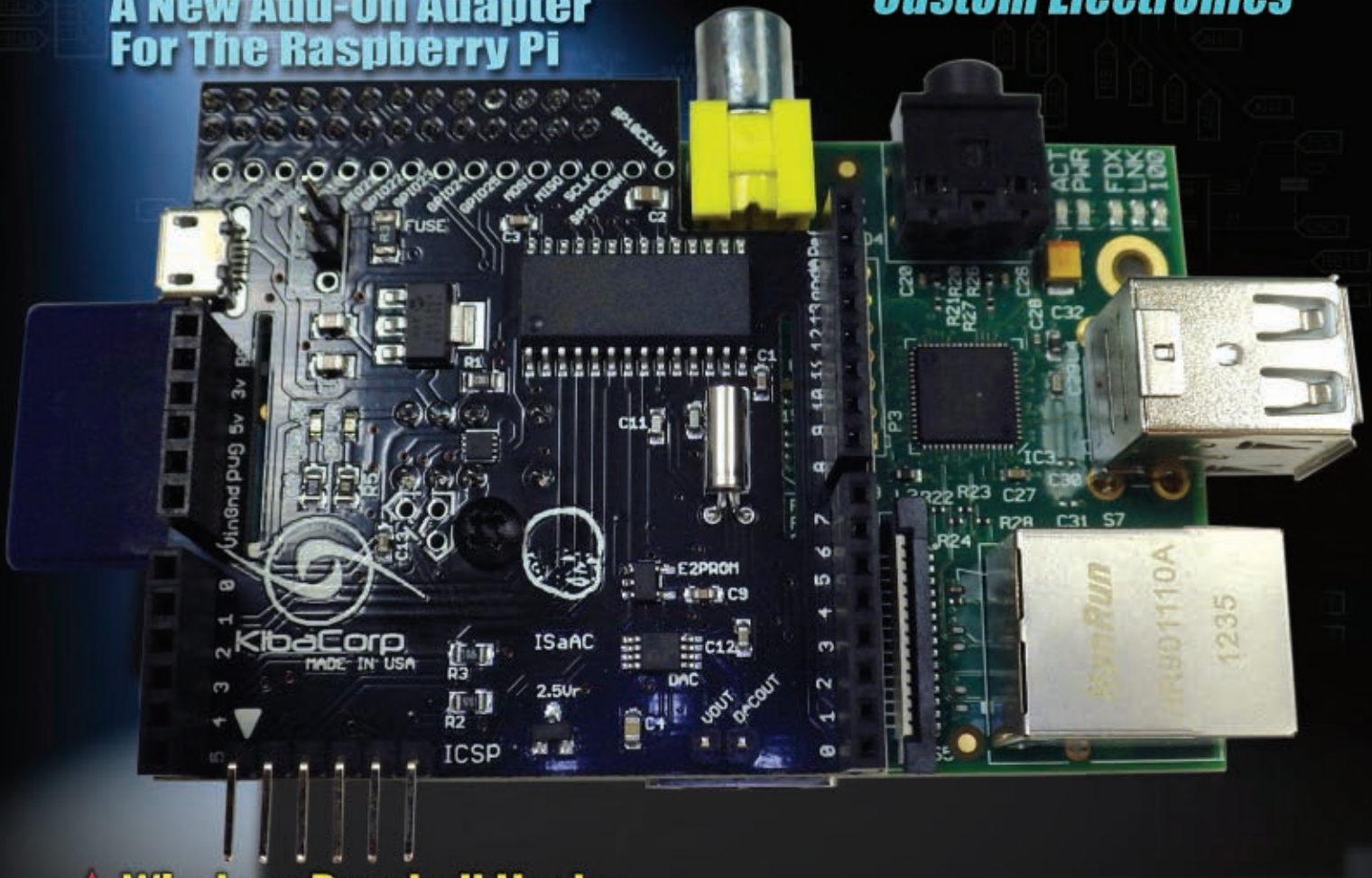
www.nutsvolts.com
August 2014

EVERYTHING FOR ELECTRONICS

Please Welcome ISaAc To The Stage

A New Add-On Adapter For The Raspberry Pi

- Digital & Analog Interfaces
- Easy To Program
- Expansion Port For Custom Electronics

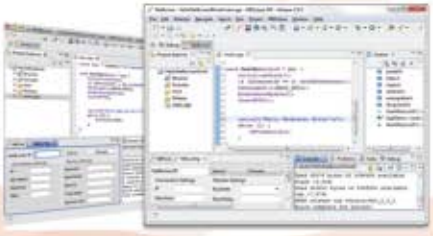


- Wireless Doorbell Hacks
- USB, LEDs, And Sensors
- Frequency Counters And Retrofitting

Ethernet Core Modules with
High-Performance Connectivity Options

- **MOD5270**
  147.5 MHz processor with 512KB Flash & 8MB RAM · 47 GPIO
  3 UARTs · I²C · SPI
- **MOD5234**
  147.5 MHz processor with 2MB flash & 8MB RAM · 49 GPIO · 3 UARTs
  I²C · SPI · CAN · eTPU (for I/O handling, serial communications, motor/timing/engine control applications)
- **MOD54415**
  250 MHz processor with 32MB flash & 64MB RAM · 42 GPIO · 8 UARTs
  5 I²C · 3 SPI · 2 CAN · SSI · 8 ADC · 2 DAC · 8 PWM · 1-Wire® interface
- **NANO54415**
  250 MHz processor with 8MB flash & 64MB RAM · 30 GPIO · 8 UARTs
  4 I²C · 3 SPI · 2 CAN · SSI · 6 ADC · 2 DAC · 8 PWM · 1-Wire® interface

Add Ethernet connectivity to an existing product, or use it as your product's core processor



The goal: Control, configure, or monitor a device using Ethernet

The method: Create and deploy applications from your Mac or Windows PC. Get hands-on familiarity with the NetBurner platform by studying, building, and modifying source code examples.

The result: Access device from the Internet or a local area network (LAN)

The NetBurner Ethernet Core Module is a device containing everything needed for design engineers to add network control and to monitor a company's communications assets. For a very low price point, this module solves the problem of network-enabling devices with 10/100 Ethernet, including those requiring digital, analog and serial control.

MOD5270-100IR.....$69 (qty. 100)
MOD5234-100IR.....$99 (qty. 100)
MOD54415-100IR.....$89 (qty. 100)
NANO54415-200IR...$69 (qty. 100)

NNDK-MOD5270LC-KIT..... $99
NNDK-MOD5234LC-KIT.....$249
NNDK-MOD54415LC-KIT.....$129
NNDK-NANO54415-KIT..... $99

NetBurner Development Kits are available to customize any aspect of operation including web pages, data filtering, or custom network applications. The kits include all the hardware and software you need to build your embedded application.

- **For additional information please visit**
  http://www.netburner.com/kits

NetBurner
Networking in One Day!

Information and Sales | sales@netburner.com
Web | www.netburner.com
Telephone | 1-800-695-6828

freescale™
Alliance Member

EasyPIC v7 with click™ boards makes perfect match for any project you are working on. Just place your click board into the mikroBUS™ host socket and it's ready to work straight away. Adding new functionality to your development board was never so easy!

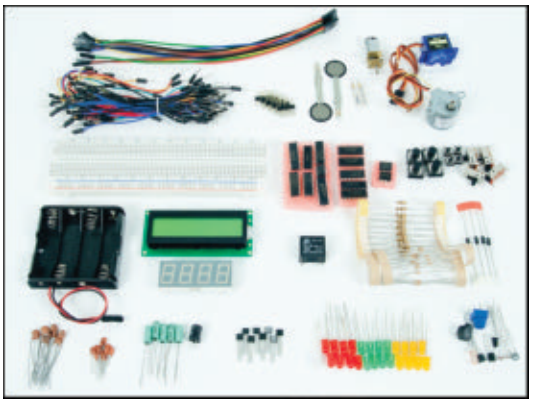
MikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS

www.mikroe.com/click/

GET IT NOW
www.mikroe.com/easypic/

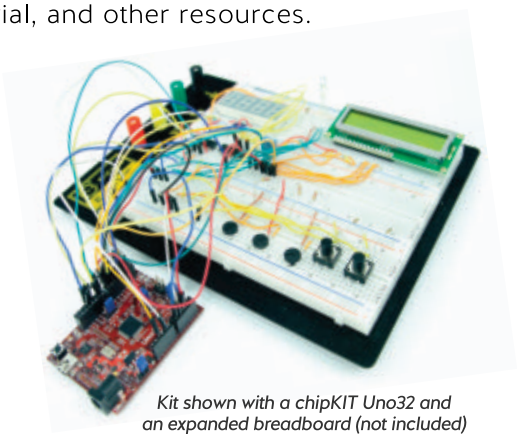
Learning by Making:

The chipKIT™ Starter Kit.



The chipKIT Starter Kit provides a wide range of sensors, actuators, and electronic components at a remarkably low cost, enabling the construction of an endless array of projects that can serve to inform beginners or empower experts. The kit has been designed for use with our chipKIT™ boards, 32-bit microcontroller-based platforms that are compatible with much of the existing Arduino® code base, reference material, and other resources.

The kit's parts have been selected to facilitate the creation of fun, useful, and entertaining projects, while at the same time facilitating the exploration of multiple disciplines, from optics to mechanics, from electrical engineering to computer science. Projects can range from as simple as blinking LEDs, to performing tracking based on optical information, to much more!



Kit shown with a chipKIT Uno32 and an expanded breadboard (not included)

The chipKIT uC32: The ideal playground for any skill set, any project.

chipKIT™ uC32

The chipKIT™uC32 is a 32-bit microcontroller platform that is compatibible with many Arduino™ code examples and shields. It is similar to our Uno32 but comes with additional memory. The uC32 can be programmed using MPIDE, a fork from the original Arduino™ IDE.

- Microchip® PIC32MX340F512H
  80 Mhz 32-bit MIPS
  512K Flash, 32K SRAM
- Designed for MPIDE
  (Can also work with MPLAB)
- 42 available I/O

Be sure to watch for learning modules based on these products and more to appear on our new educational offering:

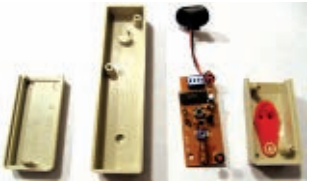
learn.digilentinc.com

chipKIT™

DIGILENT®
BEYOND THEORY

www.digilentinc.com

August 2014



Page 28

Departments

Page 68

Columns

Nuts & Volts (ISSN 1528-9885/CDN Pub Agree #40702530) is published monthly for $26.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL MAILING OFFICES. POSTMASTER: Send address changes to **Nuts & Volts, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@nutsvolts.com.

WHAT'S HOT

Test & Measurement and Other Essentials!

OSCILLOSCOPES

Rigol

RIGOL's test & measurement instruments compete with industry leaders – but more affordably: Analog & Mixed-signal Scopes to 1GHz with AWG, Function/AWGs from 20MHz-350MHz RF Signal Gens to 6GHz, Pwr Supplies, DMMs, & Spectrum Analyzers.

Owon

OWON's affordable, reliable, easy-to-use precision benchtop & handheld scopes are unbeatable in their price range. Battery powered and portable options for field use. Owon's Triple Output Power Supplies offer remote control & preset configurations.

Pico

PICO TECHNOLOGY - the world's best PC-based oscilloscopes & data acquisition equipment with the performance of good bench scopes; compact & lightweight - ideal for field work. Award-winning automotive scopes find engine & electric faults quickly.

MISC.

USB Control

Need to interface a PC USB port to other equipment? We have modules for USB-I2C with additional I/O, USB data acquisition, USB-serial (some within the connector itself!), the best and most reliable USB-serial cables, and USB isolators to eliminate ground loops.

Power Supplies

We stock numerous power supplies from simple "wall-warts" for low-voltage/ low-current needs, to more complex bench-top supplies with single or multiple outputs, programmable supplies, flexible output current/voltage supplies with multiple communication options.

3D Printers

3D printing makes product and device development faster and cheaper – rapidly build concept parts and projects. We have the latest 3D printers from industry leaders MakerBot and Afinia, as well as the best 3D printing filament available for creating high quality parts quickly!

CIRCUIT/PCB TEST

ABI Electronics

ABI ELECTRONICS - PCB test/repair: **BoardMaster 8000:** easy-use PCB test system; **JTAGMaster:** programs/tests via JTAG port; **RevEng Schematic Learning System:** generates schematics from PCBs; **SENTRY:** detects counterfeit components.

MQP

MQP ELECTRONICS – USB2.0 Analyzers /Gens, Protocol/Electrical Test equipment, offering unbeatable value/performance ratio and Vbus monitoring. GraphicUSB software offers full analysis of std USB2.0 protocol with Class Analysis options.

Cosview

COSVIEW - range of extremely affordable USB-connected inspection microscopes & stands, very useful for examining printed circuit boards and small integrated circuits for production faults or part number markings. Polarizing filter option which excludes surface glare is available.

DEVELOPING PERSPECTIVES

The DIY Differential

When shopping recently for a large LED digital clock, I was caught in a common dilemma: Do I go for the inexpensive import for $15 or spring for the $90 DIY kit? In this case, the issue was time — I didn't have time to build the kit and needed the large digit clock for an upcoming project. So, I went with the $15 option.

The Chinese-manufactured clock performed flawlessly ... for about a week. Then, the display was nothing but random LED segments. When I cracked open the case, I found nothing in the way of user-serviceable parts. Everything was soldered in place, including the main IC which looked like a spider epoxied to the motherboard. So, there went $15 plus a lot of time and trouble.

I ended up using a different time-keeping system for the project, and all was well.

After the crunch, I revisited the world of large digit LED clocks. This time, I went for the $90 kit. After three hours of soldering and a bit of sanding, the clock was ready for mounting. Although I haven't exercised the option of reprogramming the clock to, say, a countdown timer, it's only a matter of Arduino programming.

Plus, there's a small breadboard area on the clock's motherboard. Moreover, I know that if the clock suddenly dies, I can resuscitate it by replacing the failed components and reloading the Arduino program if necessary.

Is this to say that relatively expensive kits are the only way to go? No — sometimes you just have to go with off-the-shelf, affordable, and sometimes cheap options. When you do have to decide, just make an informed decision. Is there something to learn from, say, building your next clock, radio, timer, LED display, or other circuit, or is your time spent better elsewhere?

It's a personal choice, and one that depends on your level of mastery in a given area — and, of, course, budget. No need to twiddle with an LED project if you're looking to learn about digital signal processing (DSP) techniques. It's better to pick an analog-to-digital converter project.

By the way, the $90 DIY clock is still running months after the $15 clock's demise. If and when the DIY clock dies, I'm sure I'll have the means to repair it. Sure, I could keep buying $15 clocks, but I'd have to deal with the uncertainty of the cheap versions failing at the worst possible moment, and the moral implications of constantly contributing to landfills.

Keep building! **NV**

NUTS AND VOLTS
EVERYTHING FOR ELECTRONICS

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct.
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore orders only **1-800-783-4624**
www.nutsvolts.com

Subscription Orders
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

FOUNDER
Jack Lemieux

PUBLISHER
Larry Lemieux
publisher@nutsvolts.com

ASSOCIATE PUBLISHER
Robin Lemieux
robin@nutsvolts.com

EDITOR
Bryan Bergeron
techedit-nutsvolts@yahoo.com

VP OF OPERATIONS
Vern Graner
vern@nutsvolts.com

ADVERTISING SALES
Garry Moore
garry@nutsvolts.com

CONTRIBUTING EDITORS
Joe Pardue
Ron Hackett
Tom Kibalo
Robert Reed
William Pippin
Fred Eady
Lou Frenzel
Ben Kibalo
Frank Muratore

CIRCULATION DEPARTMENT
subscribe@nutsvolts.com

SHOW COORDINATOR
Audrey Lemieux

WEB CONTENT
Michael Kaudze
website@nutsvolts.com

WEBSTORE MARKETING
Brian Kirkpatrick
sales@nutsvolts.com

WEBSTORE MANAGER
Sean Lemieux

ADMINISTRATIVE STAFF
Debbie Stauffacher
Re Gandara

READER FEEDBACK

Generating Interest

I liked Robert Reed's June 2014 article on the 150 MHz RF signal generator. I am going to propose it to our club as a build project as we may have about 6-10 members interested in rolling these. To that end, I may end up designing the RF board on a ground plane style PCB to make it easier to solder.

A few points worth mentioning:

1. Would there be a newer alternative to the components listed? It may not be as flexible, but one-of-a-kind parts get tricky and I have been

Continued on page 72

■ BY RON HACKETT
www.jrhackett.net

Build an Auto-Off Continuity Tester

At the end of the previous Primer installment, I requested some reader feedback, asking the following questions:

- **Do you want to continue our PICAXE-Pi explorations?**
- **Or, would you rather intermix the Pi coverage with some PICAXE projects?**
- **Or, have you had enough Pi for now, and want to return to pure PICAXE?**

So far, only five readers have emailed me with their preferences: two said they would like to continue with the PICAXE-Pi projects; two preferred to return to pure PICAXE projects; and one said that whatever I decide to write about is fine with him. As a result, I still don't have a clear sense of the content that readers would prefer to see here. Of course, you can still email your preferences to me at Ron@JRHackett.net, but for the time being, I'll just continue to pursue topics that interest me!

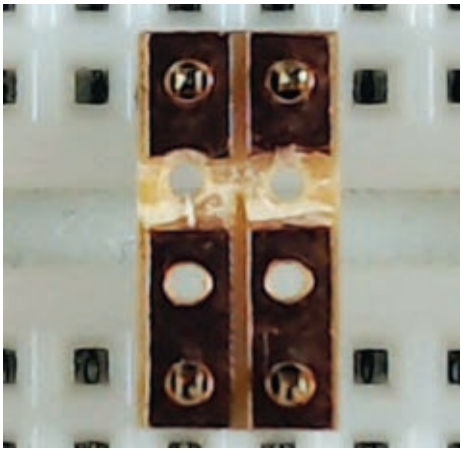
This month, we're going to take a look at the CPC1002N which is an SMD solid-state relay (SSR) that contains an optically coupled MOSFET transistor. As I mentioned last time, the CPC1002N is capable of switching up to 700 mA of DC current at voltages as high as 60V, so it's a useful device to keep in mind for a variety of PICAXE and/or Pi projects. However, because the CPC1002N is an SMD device, it's not very "breadboard friendly." So, the first thing we need to do is remedy that problem.

However, before we begin our work with the CPC1002N, I want to give you another update on the PICAXE-Pi printed circuit boards (RazzPi-LCD and RazzPi-20) that I also mentioned last time. There were a couple of minor issues with the first batch of boards that I received, so I decided to make the necessary modifications and have another batch produced. As a result, the two boards are not yet available on my website; hopefully it won't take too long to get the second batch.

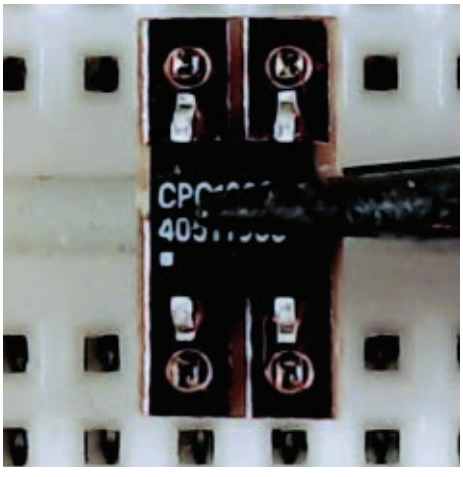
Making the CPC1002N Breadboard Friendly

In order to use the CPC1002N in a breadboard circuit, all we need to do is solder it onto a small piece of stripboard, along with a couple of two-pin male headers. The necessary circuit is so simple that it doesn't require a stripboard layout to construct; a couple of photos will suffice.

Figure 1 shows the small piece of stripboard that we need (two traces with four holes each). In the photo, you can see that the stripboard is mounted on two two-pin male headers with its traces facing up. Also, both traces are cut at one of the middle holes so that all four pins will be electrically isolated from each other.

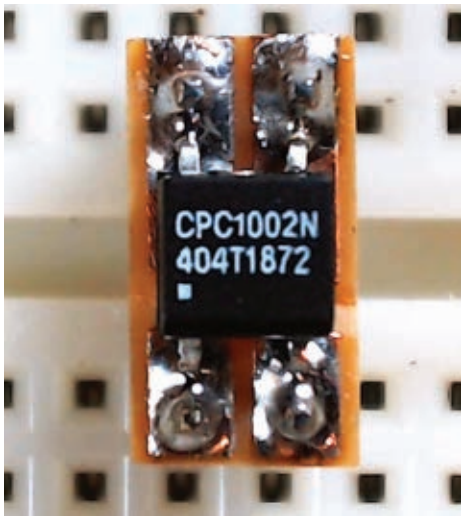


■ FIGURE 1. Stripboard base for CPC1002N.

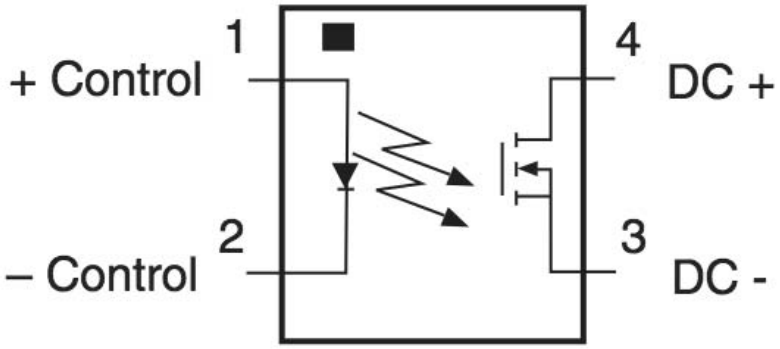


■ FIGURE 2. CPC1002N clamped to base.

In **Figure 2**, a CPC1002N is being held in place by tweezers in preparation for soldering each of the SSR's pins to the stripboard and one of the four header pins. In the photo, you can see the SSR's white "Pin 1" marker just below the tweezers on



■ FIGURE 3. CPC1002N soldered to base.



■ FIGURE 4. Schematic and pinout for CPC1002N.

the left. The photo in **Figure 3** shows the CPC1002N after it has been soldered in place. That's all there is to it! If you decide to carry out the experiments that we're about to discuss, all you need is a couple of CPC1002N devices which are available on my website along with the CPC1002N datasheet. If you don't already have an extra piece of stripboard and the necessary male headers, they are also available.

Experiment 1: Testing the CPC1002N

For our first experiment, we're just going to connect an LED to the output side of the CPC1002N and use the SSR to control the state of the LED. Of course, we don't need an SSR to blink an LED; any PICAXE output pin is all we need for that purpose.

However, as I mentioned earlier, the CPC1002N can switch as much as 700 mA of DC current at voltages as high as 60V, so it can also be used to drive small DC motors, fans, solenoids, etc., which makes it a useful device for a variety of PICAXE (and Pi) projects.

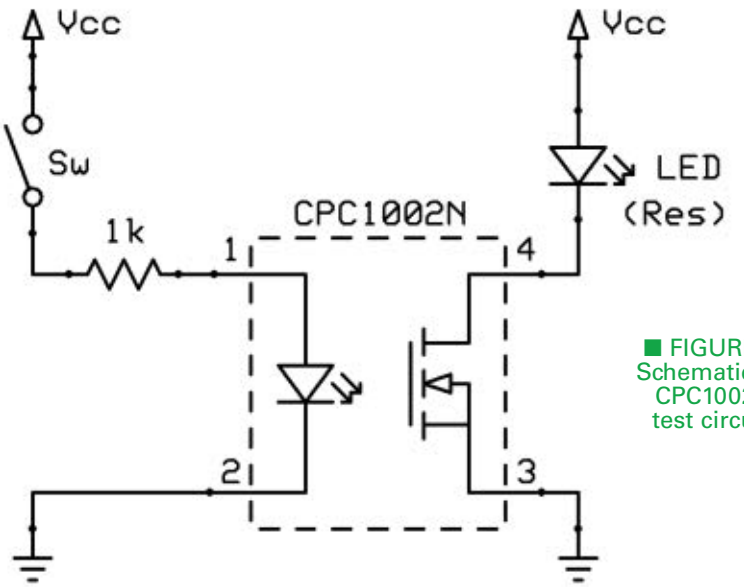
Figure 4 is the schematic and pinout of the CPC1002N. Pin 1 is connected to the control voltage and pin 2 is the control ground connection. Note that both the input and output pins are polarized; pins 1 and 4 must be connected to positive DC voltages. The internal IR LED is current controlled and requires a minimum of 2 mA to operate. Also, the LED has a typical voltage drop of 1.2V, and requires a current-limiting resistor in the input circuit. (See the CPC1002N datasheet for more details.)

Figure 5 is the schematic of the simple circuit that we'll use to test the CPC1002N stripboard, and **Figure 6** is a photo of my breadboard setup for the test. As I mentioned earlier, the typical voltage drop across the CPC1002N internal LED is 1.2V. Since I'm using a +5V power supply, the remaining 3.8V is dropped across the current-limiting resistor.

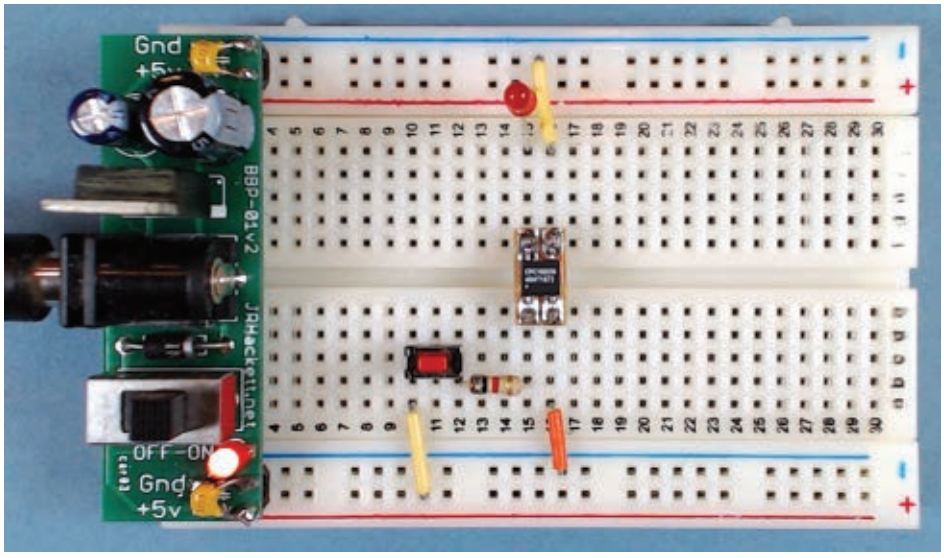
I chose to use a 1K resistor because it results in a 3.8 mA input current (3.8V / 1K = 3.8 mA) which is comfortably above the minimum 2 mA current that's required to light the SSR's internal LED. If you use the CPC1002N with a different supply voltage, you will need to determine an appropriate value for the current-limiting resistor.

Also, note that I used a resistorized LED in the output circuit. If you use a non-resistorized LED, don't forget to include a current-limiting resistor in the output circuit, as well.

Using either **Figure 5** or



■ FIGURE 5. Schematic for CPC1002N test circuit.



■ FIGURE 6. Breadboard setup for test circuit.

Figure 6, set up your breadboard circuit and test your CPC1002N stripboard. Whenever you press the switch, the LED should light and it should turn off as soon as you release the switch. If not, you will need to troubleshoot the soldering connections on your CPC1002N stripboard circuit.

The most likely culprit would probably be an accidental short between two of the SSR's pins. When you have your CPC1002N board functioning correctly, we're ready to move on to our next experiment.

Experiment 2: A Simple Auto-Off Circuit

In many battery-powered projects, it's often helpful to be able to turn off the project's power via its own software. For example, in a battery-powered count down timer, it would make sense for the timer to automatically power-down when the alarm has been sounded for a specified amount of time.

The CPC1002N provides an easy way to implement that capability in any program, and this experiment is a simple demonstration of one way to do so. We're again going to just blink an LED to demonstrate the process, but the PICAXE processor could be controlling a variety of I/O devices when the program disconnects the power.

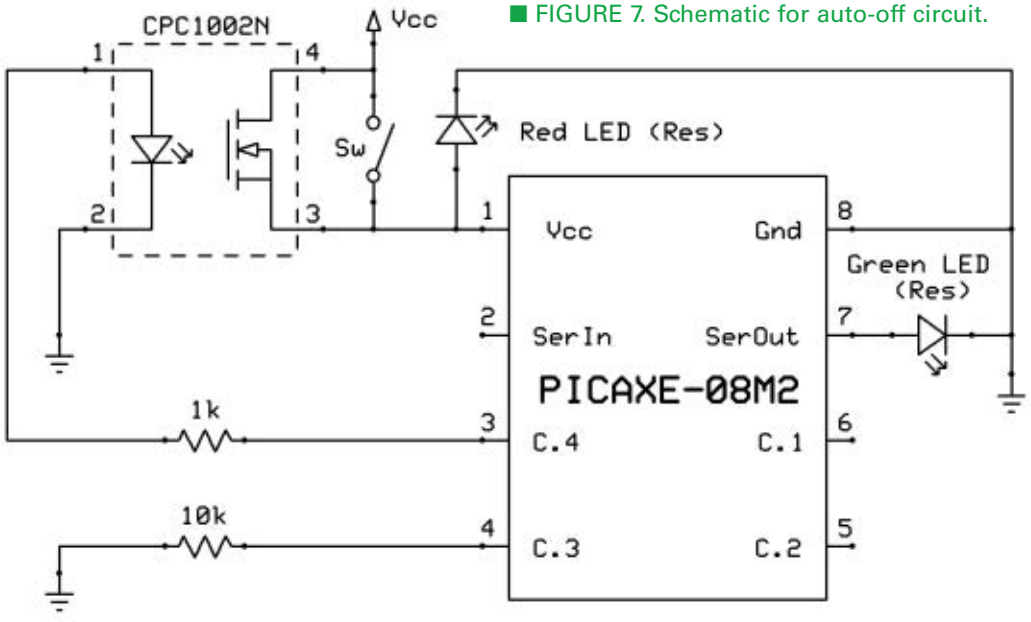
Figure 7 presents the schematic for this experiment. The most important part of the hardware setup is the arrangement of the CPC1002N and the normally open pushbutton switch in the upper-left corner of the diagram.

As you can see, the pushbutton is connected in parallel with the SSR's two output pins. Vcc is connected to one side of the pushbutton and to pin 4 on the SSR, which is the positive connection for the relay's output. (Refer back to **Figure 4**.)

When power is first connected at Vcc (we'll again use +5V), the switch is open (its normal state), and the relay is not active because no current can flow from pin C.4 on the 08M2 until the processor is powered. In other words, on the SSR, pins 4 and 3 are not yet connected. (Remember, only a current flow into pin 1 of the SSR can activate the relay.) As a result, even though power is connected at Vcc, the 08M2 processor is **not** powered and the red LED is **not** lit at this point.

However, pressing the pushbutton switch immediately applies power to pin 1 of the processor and (as we will soon see when we discuss the software for this experiment) the very first programming instruction is *high C.4*, which lights the SSR's internal LED and

■ FIGURE 7. Schematic for auto-off circuit.



activates the relay which provides a second connection between Vcc and pin 1 of the processor.

Consequently, when the pushbutton switch is released, the SSR continues to power the 08M2. (Yes, it is possible to press the pushbutton rapidly enough to defeat this arrangement, but you would never want to do that, would you?)

Before moving on to the hardware setup for this experiment, I need to make one final point about the schematic. You probably noticed that I did not include the standard PICAXE programming connections, however, they certainly need to be in the circuit!

I used one of my programming adapters, but you can use whatever programming arrangement you want.

Figure 8 is a photo of my breadboard setup for the experiment. You may want to trace the connections to see how they match up with the schematic.

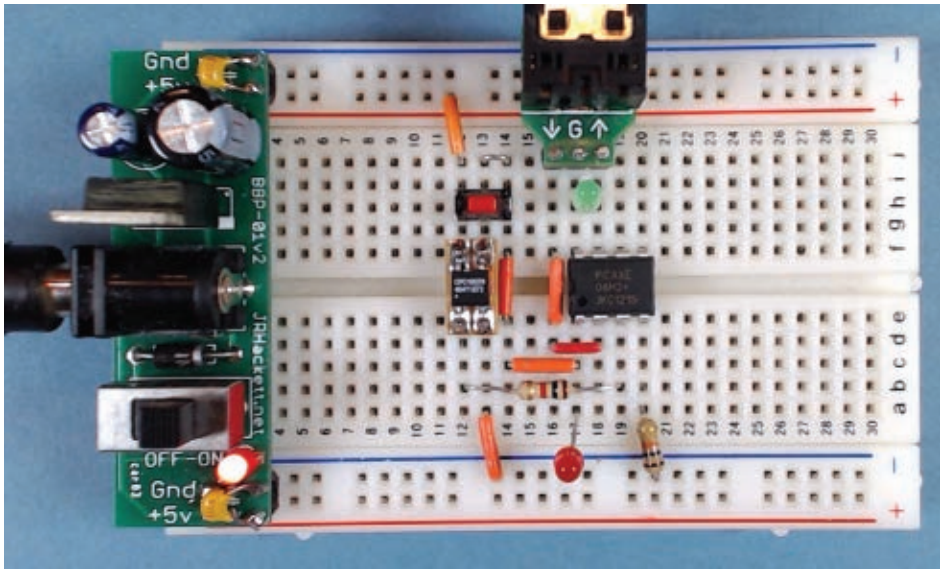
When you've completed your own breadboard circuit, we can take a look at the software for this experiment (*AutoOffDemo.bas*) which is available for downloading from the article link. (While you're there, also grab the other files that we'll use this month.)

The program itself is fairly self-explanatory. As I mentioned earlier, the very first programming statement (*high SSR*) energizes the SSR so that the circuit remains powered after the pushbutton is released.

The *for/next* loop is trivial; it simply gives the program something to do before it powers itself down by executing the final *low SSR* command.

When you've constructed your breadboard circuit and are ready to download the *AutoOffDemo.bas* program to the 08M2, you will encounter a minor problem. Since the processor is not powered, the PICAXE Editor will refuse to download the program!

You can solve this problem by inserting a temporary jumper wire between pin 1 of the 08M2 and the Vcc connection point. However, I think it's even easier to just press and hold the pushbutton switch throughout the program download.



■ FIGURE 8. Breadboard setup for auto-off circuit.

As soon as the green LED starts to blink, the CPC1002N has connected the Vcc line to the processor, and you can allow the switch to open. The green LED will continue to blink a total of seven times, with the red LED remaining lit throughout the entire process.

Shortly after the last blink of the green LED, the red LED will turn off, indicating that power has been disconnected from the processor. Whenever the red LED is off and you briefly press the switch, the program will run again and turn itself off when the blinking is finished.

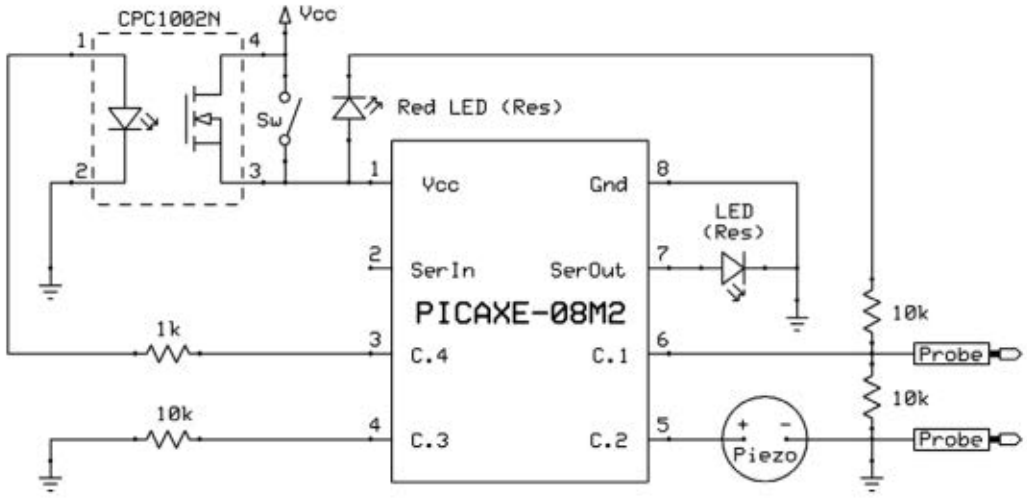
As we have seen, the CPC1002N SSR makes it easy to add an auto-off feature to any PICAXE project that would benefit from that functionality. All we need to do is attach the same configuration of the SSR and a normally open pushbutton switch to the Vcc pin of the processor, begin the program with a *high SSR* statement, and (when all processing tasks have been completed) disconnect the power with a *low SSR* statement. That's exactly what we're going to do next.

Experiment 3: A Simple Auto-Off Continuity Tester

I've been using the same digital multimeter (DMM) for close to 20 years now. It has all the functionality that I need, except for one small detail: It does not have an auto-off feature. Since I often forget to turn off the DMM when I'm finished using it, I spend a lot more on batteries than I should.

For some reason, I'm especially forgetful when I've been using the DMM as a continuity tester. To solve this problem, I decided to construct a simple PICAXE-based auto-off continuity tester.

Figure 9 shows the schematic of the breadboard version of the project. If you compare it to the schematic we just discussed (**Figure 7**), you will see that there are only three additions: a piezo is connected to pin C.2 for audible output; a 10K/10K voltage divider is connected to pin C.1; and two "probes" have been connected to the voltage divider. (For this experiment, the "probes" are actually two long pieces of jumper wire pretending to be



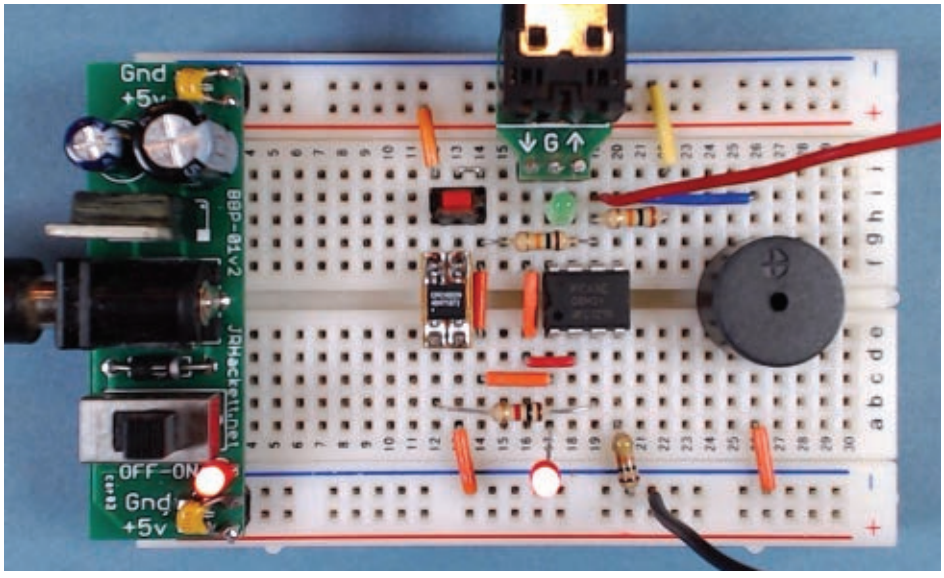
■ FIGURE 9. Schematic for continuity tester (v1).

probes.) One probe is connected to the mid-point of the voltage divider, and the other probe is connected to ground.

My breadboard setup for this experiment is shown in **Figure 10**. To construct this setup, I just added the two 10K voltage divider resistors, the two probes, and the piezo to my breadboard setup from Experiment 2. In the photo, you can see the two probes (the long red and black jumper wires extending to the right). When you've completed your breadboard setup, we can continue on to the software for this experiment (*ConTester.bas*), where we'll see how the probes function.

When you're ready, print out a copy of the program for reference in the following discussion. As we've done before, the following numbers refer to the corresponding numbers at the end of several of the program lines.

1. I do realize that our two "probes" aren't really multimeter probes but you can probably guess where we're headed, and I want to use the same program when we get there.



■ FIGURE 10. Breadboard setup for continuity tester.

2. I'm not sure if we've done this before, but we're going to use the *pwmout* command to drive the piezo. Using the *sound* command is the more usual approach to producing a "beep" on a piezo, but it completely ties up the processor until the beep has finished playing. We want the main loop to execute as fast as possible, and *pwmout* does the job.

3. All PICAXE variables are initialized to zero whenever a program first runs, so this *if/then* statement causes the LED on pin C.0 (*SerOut*) to blink the first time through the main loop.

4. Here, we increment the *alive* variable each time through the main loop. All byte variables "roll over" after 255 (i.e., their value becomes 0 again). As a result, the LED will blink once every 256 times through the main loop. On my setup, the LED blinks roughly every 3.2 seconds which provides a good reminder that the program is still running.

5. One probe is connected to the middle of the 10K/10K voltage divider and the other probe is connected to ground. Therefore, if the two probes are touching two points in a circuit that are in no way connected to each other, *ADCval* will be approximately mid-scale (i.e., 128 for a byte variable).

If the two probes are touching two points in a circuit that are directly connected to each other, *ADCval* will be (theoretically) 0 because the probe at pin C.1 is effectively shorted to the probe at ground. So, it will also be at ground. Of course, all wires and traces have some minute amount of resistance which is why we're using "less than 3" rather than "equal to 0" for our comparison.

Finally, if the two probes are touching two points in a circuit that

are on opposite sides of a resistor, that resistor will be in parallel with the lower 10K resistor in the voltage divider circuit As a result, *ADCval* will be somewhat lower than mid-scale. In fact, if that resistor happens to have a value of 2Ω or less, it would — in effect — fool the program into reporting a short. However, I've never used a resistor that small in a PICAXE project, so I'm not too worried about that possibility.

(In defense of our little continuity tester, a commercial DMM would also be fooled into beeping; the difference is that a commercial DMM would also report the resistance that's being measured.)

6. As you may remember, the PICAXE *time* variable is a built-in system word variable, which means that it does not need to be declared in a program. Also, like all variables, it's automatically initialized to zero whenever a program starts running. We're using the *time* variable to disconnect the power from the 08M2 whenever a short has not been detected for more than 10 minutes (600 seconds). Therefore, each time we do detect a short, we reset time to 0, and start the count again.

7. Whenever a short is detected, we also beep the piezo.

8. On the other hand, if a short is not currently detected, we silence the piezo.

9. Here, we test to see if 10 minutes have passed without a short being detected. If so, we disconnect the main power. If not, we continue looping. If you prefer a longer or shorter auto-off period, just change the value from 600 to whatever you prefer.

Since you waded through that long-winded explanation of the program, you deserve a reward; download it to your breadboard setup and give it a try.

■ FIGURE 11. Continuity tester mounted in battery holder.

Constructing a "Real" Continuity Tester

Our little continuity tester experiment worked so well for me, that I couldn't resist constructing a stripboard version of the project. (You knew that was going to happen, didn't you?) For convenience, I decided to use three AA alkaline cells to power the project. Rather than using an enclosure, I designed the stripboard circuit so that it would fit into one of the battery slots in a four AA cell battery holder (RadioShack #2700391).

To get a clear idea of where we're headed, you can take a look at the completed project shown in **Figure 11**.

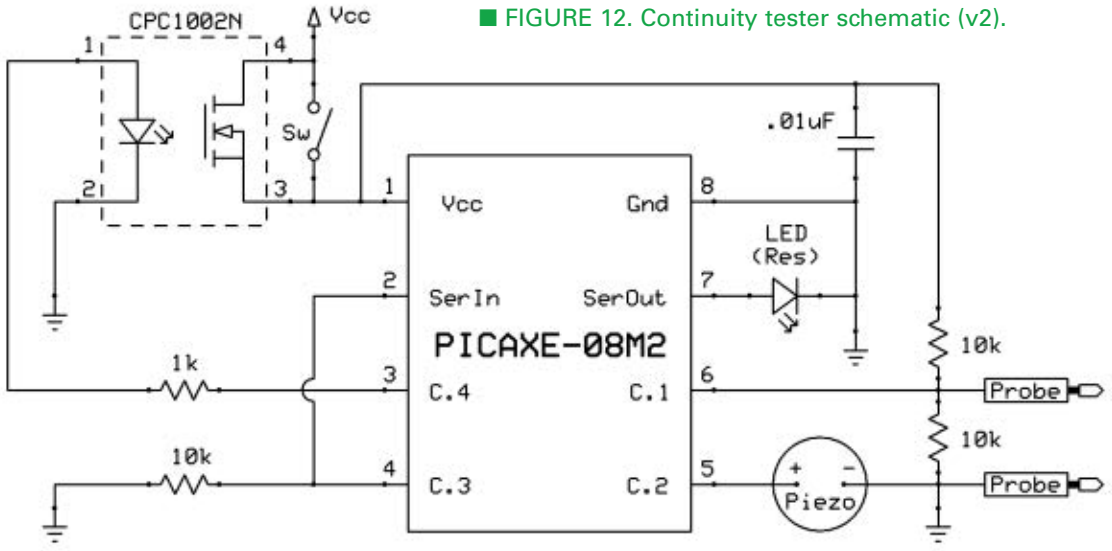
Getting the stripboard to fit inside the battery holder turned out to be a bit of a challenge. Even so, I don't regret my decision because the alternative of using a standard project enclosure that could accommodate three AA cells and the stripboard would have at least doubled the expense. (Also, I really like how the completed project looks!)

After finishing this article, if you decide you would prefer an alternative approach, it would certainly be easy to make the stripboard a little larger than mine, and then just mount the battery holder, the stripboard, and two banana jacks on a small piece of wood or plastic.

If you decide to take that approach, don't forget that you can't power a PICAXE circuit with four AA cells; the voltage — which can be greater than +6V — would most likely destroy the PICAXE processor. In the small space that I had allowed myself, there was no room for a programming connector, so I just used the same 08M2 that we already programmed in Experiment 3. The program is also the same as that of Experiment 3, but if you want to make any changes to it you can just re-program the 08M2 on the breadboard circuit and then insert it into the IC socket on the stripboard.

The schematic for this project is in **Figure 12**. It's very similar to the schematic for Experiment 3, but it includes the following three minor modifications:

- My breadboard power supplies all have bypass caps on the power rails, so I don't usually add another one to the breadboard circuit. However, it's always a good idea to include a bypass capacitor in any PICAXE stripboard circuit.
- I removed the LED that was connected to pin 1 of the 08M2 to conserve battery power. The blinking LED on the *SerOut* line (pin C.0) clearly indicates when the program is running. When it stops blinking, power has been disconnected from the circuit.
- All the programming adapters that I use include the necessary circuitry to pull the *SerIn* line to ground, which is required for any PICAXE processor to function correctly. As I've already mentioned, there wasn't enough room to include a programming connector on the stripboard, so I needed a way to pull *SerIn* to ground. I first tried to add a resistor between *SerIn* and ground,



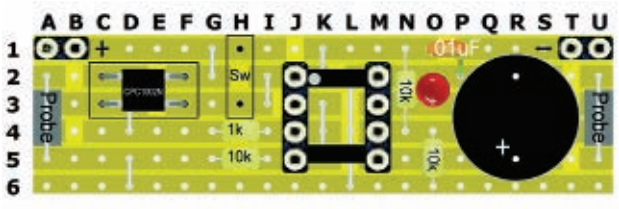
■ FIGURE 12. Continuity tester schematic (v2).

but that turned out to be surprisingly challenging. I finally decided on a simpler approach. I directly connected *SerIn* to pin C.3 which is already pulled to ground by a 10K resistor. As you know, *SerIn* and pin C.3 are both fixed as inputs, so directly connecting them presents no risk of an accidental short.

The stripboard layout for the continuity tester is presented in **Figure 13**; a large version of the layout is available for downloading from the article link. **Figure 14** is the complete Parts List for the project. Except for the 08M2, the battery holder (which is RadioShack #2700391), the batteries, and the multimeter probes, all parts are available on my website.

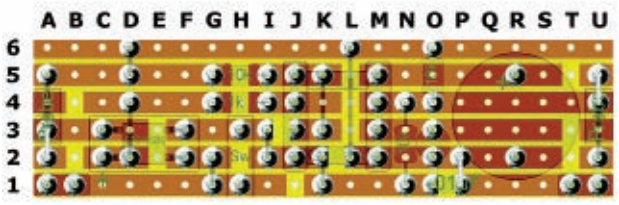
Before we continue, there are four points that I need to clarify:

1. The rectangular black outline around the CPC1002N indicates that we're using the same small stripboard assembly that we used in our earlier experiment, and are soldering it in place on top of the main stripboard. (Refer to the photo of my completed circuit shown in **Figure 15** and you will see what I mean.)



Top View

■ FIGURE 13. Stripboard layout for continuity tester.



Bottom View

2. If you look back at the layout in **Figure 13**, you can see that row 6 is slightly narrower than the other rows. That's because six full rows won't quite fit into the slot in the battery holder; row 6 needs to be sanded a little, until the board will fit snugly into the battery holder. I think it's better to wait before sanding and fitting the board until the three connections in row 6 have been soldered. Of course, you need to be very careful not to sand so far that you sever the leads at holes D6, L6, or O6!

Again, refer to the photo in **Figure 15** to see how close I sanded to those three leads. (This is one of the reasons you may prefer not to fit the stripboard into the slot in the battery holder.)

3. When I first started thinking about the stripboard version of this project, I happened to have the RadioShack battery holder and set of their banana jacks in my stockpile of miscellaneous parts. (In fact, it was those two parts that gave me the idea to squeeze the stripboard into the battery holder slot.)

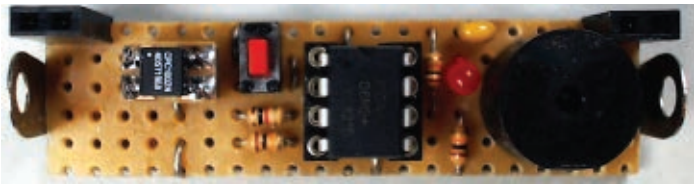
When I designed the stripboard layout, I decided to bend the two metal tabs that came with the banana jacks so that they extended underneath the board, and solder them on the bottom of the board so their connections matched that of the schematic. However, that decision led to two major frustrations.

First, it was very difficult to make the bend in each tab so that everything lined up properly. Second, it required so much heat to solder each tab, that I melted both of the two-pin female headers, and had to desolder and replace them.

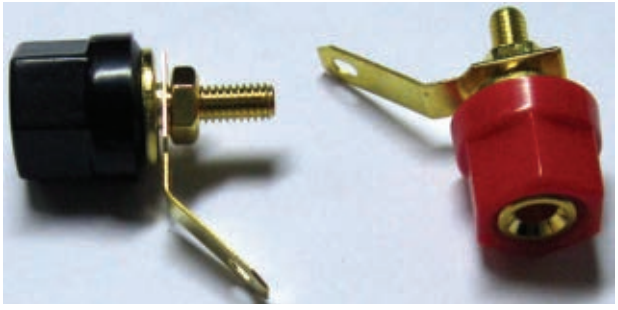
It wasn't until I finally managed to get everything fitted and working correctly, that I realized it would have been much easier to bend the tabs so that they ran along the top of the stripboard rather than the bottom. We'll see exactly how to do that when we assemble the board, but I wanted to mention it now in case you spotted the discrepancy between **Figures 13** and **15**. (In **Figure 13**, you can see that the metal tab is on the top of the board with a jumper spanning it. However, in the photo of

Batteries, three Alkaline AA cells**
Battery Holder, RadioShack #2700391**
Banana Jacks (set of two)
Capacitor, 0.01 µF
CPC1002N SSR (see text for mounting)
Headers, female, two pins (two pieces)
IC Socket eight-pin, machined
LED, 3 mm, red, resistorized
Multimeter Probes (set of two)**
PICAXE-08M2 Microcontroller**
Piezo
Resistor, 1K, 1/6 watt
Resistors, 10K, 1/6 watt (three pieces)
Stripboard, six traces with 21 holes
Switch, reset
Not available at **www.JRHackett.net

■ FIGURE 14. Continuity Tester Parts List.



■ FIGURE 15. Completed continuity tester.



■ FIGURE 16. Banana jacks.

Figure 15, the tab is clearly **not** on the top of the board!)

4. When I finally had my stripboard circuit functioning correctly, I turned my attention to assembling the parts list for the project, and was not happy to learn that RadioShack no longer carries the banana jacks that I used! (Did I hear someone say, "Look before you leap!") Fortunately, I was able to locate a similar set of jacks that should work fine in the project (see **Figure 16**).

Okay, let's move on to actually building this thing! We'll begin with the battery holder. The following list of instructions presumes that you have oriented the battery holder so that the power and ground wires are extending from the lower right side.

❑ Using a 3/32" drill bit, drill out the lowest rivet on both the right and left sides of the holder. On the right side, this will disconnect the red power wire; on the left side, it will disconnect the spring and black wire, but the black wire will still be connected to the loose spring.

❑ Snip the spring from the black wire on the left side, keeping as much of the wire as possible. (We'll need it later!)

❑ Pull the left side black wire out of the slot in the bottom of the battery holder and pass it back up through the hole in the upper left corner of the bottom of the holder where the wire is connected to the top rivet on the left side.

❑ Thread the black wire on the right side back into the holder and through the top slot on the right side so that the wire ends up in the right side of the top battery slot. It's also possible to push the wire back through the hole from which it emerges, drill a hole near the bottom right corner of each of the two sections of plastic that separate the batteries, and route the wire up to the top battery slot inside the holder. That's the way I did it, but running it on the outside would certainly be easier.

❑ Either way, you end up with both black wires inside the top battery slot. The black wire on the left side (+V for the three remaining batteries) will be inserted into either position of the two-pin female header at the top left of the stripboard, and the black wire on the right side (ground for the three remaining batteries) will be inserted into either position of the two-pin female header at the top right of the stripboard. (Since both wires are stranded, the ends will need to be tinned before inserting them into the headers.)

❑ Use a 1/8" drill bit to enlarge the two holes from which you removed the rivets so that the small bolt on each banana jack fits through the hole. (Don't install the jacks yet.)

❑ You may also want to snip off the curved portions of plastic on the bottom edge of the holder to make it easier to insert the stripboard (see **Figure 11**).

When you've finished preparing the battery holder, we're ready to assemble the stripboard circuit. However, we don't have enough space left this month to include the list of assembly instructions, so it's available at the article link. You may be surprised at the length of the list; as I mentioned earlier, squeezing the stripboard into the slot of the battery holder turned out to be more of a challenge than I had anticipated!

If the entire process seems too involved, you could easily choose the option I mentioned earlier. Just enlarge the stripboard a bit, so you can mount everything on a small piece of wood or plastic. Either way, our continuity tester is a handy device to have available.

What's Next?

Well, I'm not sure! I'm still hoping for more reader feedback. If you haven't emailed me your answers to the questions I posed at the beginning of this article yet, I would really like to hear from you. In the meantime, I'll put on my thinking cap and see what I can come up with. See you next time ... **NV**

■ BY FRED EADY

Finishing Touches

So far, we have managed to awaken our FT800 and post some words and a button on the screen. We still have some unfinished business to settle. We also have some additional exploring to do. With that, let's get started.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/august2014_DesignCycle**.

Last Time

I left you with the code that produced the output you see in **Photo 1**. In Ricky Ricardo's world, I now have some "splaining" to do.

■ Photo 1. That OK button is just begging to be touched. However, we haven't climbed high enough on the mountain of EVE knowledge to make that happen just yet.

Once the FT800 is initialized, the host microcontroller has the ability to manipulate the FT800 under application control. The FT800 wants to see Display Lists. A Display List is made up of basic graphic primitives such as points, lines, and bitmaps. Display Lists are swapped in and out to interface with the human finger pointer. A selected Display List is shown while the next one is being constructed. A Display List always begins with a clear screen operation and ends with a swap. Thus, changes to the display are the result of multiple Display List swaps that are dictated by the application.

Drawing stuff with lines and points can be a tedious undertaking. To make display construction a bit easier, the FT800 supports widgets through its Graphics Engine. Widgets are lines, points, and bitmaps that are combined to form buttons, gauges, and text.

The FT800 Graphics Engine operates within a 4 KB ring buffer which begins at address 0x108000. The location 0x108000 is identified as *RAM_CMD* in the firmware.

Ring buffers operate by moving data around using head and tail pointers. The Graphics Engine is no different. However, instead of heads and tails, the Graphics Engine uses values stored in *REG_CMD_WRITE* and *REG_CMD_READ*. When *REG_CMD_WRITE* is equal to *REG_CMD_READ*, the ring buffer is idle. As commands are written to the ring buffer, *REG_CMD_WRITE* is incremented. The Graphics Engine detects the difference between *REG_CMD_WRITE* and *REG_CMD_READ*, and processes the commands while simultaneously incrementing *REG_CMD_READ*.

When *REG_CMD_READ* catches up to the value of *REG_CMD_WRITE*, the ring buffer goes back to the idle state. Thus, the ring buffer stop point is defined as the value of *REG_CMD_READ*, while the starting point is defined as the value of *REG_CMD_WRITE*. The ring buffer pointers are easily manipulated using the *rd32* function we defined earlier:

```
unsigned int cmdBufRd,cmdBufWr;
unsigned short cmdOffset;

cmdBufRd = rd32(REG_CMD_READ);
       //get Graphics Engine stop point
cmdBufWr = rd32(REG_CMD_WRITE);
       //get Graphics Engine start point
cmdOffset = cmdBufWr;
       //store the ring buffer start offset
```

Note that we assigned a value to the variable *cmdOffset* without checking the ring buffer for an overflow condition. In this case, we won't be writing enough Display List data to remotely approach the 4092-byte ring buffer maximum size. When writing multiple Display Lists, we must calculate the ring buffer free space. Here's the free space calculation code:

```
if ((4096 - (cmdBufWr - cmdBufRd)) > 4)
{
       // Load the ring buffer
}
```

The *cmdOffset* value is normally measured in multiples of four, which is the length of a Display List command. However, it may be necessary to compute an offset using numbers other than multiples of four. So, to accommodate any offset value and calculate a valid new offset value, we include this function in our Display List application:

```
unsigned short inccmdOffset(unsigned short
curIndex, unsigned char offset)
{
     unsigned short newIndex;
     newIndex = curIndex + offset;
     if(newIndex > 4095)
     {
          newIndex -= 4096;
     }
     return newIndex;
}
```

With the addition of our new *inccmdOffset* function, we can now mix in the *wr32* function and form a new function that allows us to insert commands into the ring buffer:

```
void cmd32(unsigned int ramcmd32)
{
     wr32(RAM_CMD+cmdOffset,ramcmd32);
        //0x108000 + cmdOffset, cmd
     cmdOffset = inccmdOffset(cmdOffset,4);
        //compute new ring buffer offset
}
```

Now that we have established the ring buffer pointers and have a function that writes commands and checks the ring buffer pointers, we can assemble a Display List. We will begin our List in the traditional manner:

```
cmd32(CMD_DLSTART);
       //issue Display List start command
cmd32(CLEAR_COLOR_RGB(0,0,0));
       //clear to black
cmd32(CLEAR(1,1,1));
       //clear color, stencil & tag
       //buffers
```

Before we attempt to put something meaningful on the display, let's get out of the truck and take a look at what it takes to put some simple text on the screen. At the very least, we will need to know how to position the text in the display's XY plane. Fonts and colors may also have to come into play.

Recall that in the previous discussion, we employed the services of the FT800 Editor to provide coordinates and font/color options. I again called upon the FT800 Editor to retrieve the syntax for the command needed to draw text. The *CMD_TEXT* command syntax provided by the FT800 Editor is outlined in **Screenshot 1**.

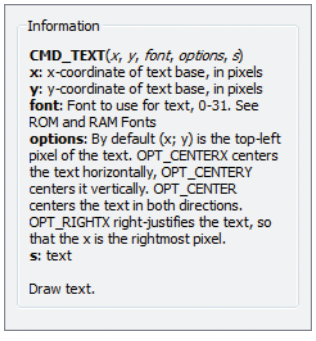
Armed with this information, I was able to construct a *drawText* macro that is based on the 16-bit version of the *cmd32* function (*cmd16*). To handle its text argument, the *drawText* macro also needs some help from the *cmdStr* function which is based on the *wrStr* function:

```
unsigned char wrStr(unsigned int addr, unsigned
char* ramstr)
{
     unsigned char i,len;

     addrBuf[0] = 0x80 | (addr >> 16);
       //parse 24-bit address
     addrBuf[1] = addr >> 8;
     addrBuf[2] = addr;

     len = strlen(ramstr);
     CSlo;
     dummybite = SPI2BUF;
       //clear BF

     for(i=0;i<3;++i)
       //send address
     {
          SPI2BUF = addrBuf[i];
            //write byte to SSPBUF register
          while(SPI2STATbits.SPIRBF == 0);
            //wait until bus cycle complete
          dummybite = SPI2BUF;
          while(SPI2STATbits.SPIBUSY == 1);
     }
     for(i=0;i<strlen(ramstr)+1;++i)
       //send string
     {
          SPI2BUF = ramstr[i];
           //write byte to SSPBUF register
          while(SPI2STATbits.SPIRBF == 0);
          //wait until bus cycle complete
          dummybite = SPI2BUF;
          while(SPI2STATbits.SPIBUSY == 1);
     }
     for(;i%4>0;i++)
       //spill out the padding chars
     {
          SPI2BUF = 0x00;
           //write byte to SSPBUF register
          while(SPI2STATbits.SPIRBF == 0);
                //wait until bus cycle complete
             dummybite = SPI2BUF;
             while(SPI2STATbits.SPIBUSY == 1);
     }
     CShi;
     return(i);
```



■ Screenshot 1. The FT800 Editor not only assists in writing Display Lists, it is also a good source of command syntax information.

As you can see, the *wrStr* function presents an address, buffered text, and pad characters to the FT800. The number of characters transferred via the SPI portal (*i*) is returned to the caller. As mentioned earlier, the *wrStr* and *inccmdOffset* functions combine to form the *cmdStr* function:

```
void cmdStr(unsigned char* ramcmdx)
{
     unsigned char rc;
     rc = wrStr(RAM_CMD+cmdOffset,
     ramcmdx);
```

```
      //0x108000 + offset, cmd
    cmdOffset = inccmdOffset(cmdOffset,rc);
      //compute new offset
}
```

The *drawText* macro is starting to shape up. Note that only two bytes are used in the *cmd16* offset calculation versus the four bytes used in the *cmd32* function:

```
void cmd16(unsigned short ramcmd16)
{
    wr16(RAM_CMD+cmdOffset,ramcmd16);
      //0x108000 + offset,cmd
    cmdOffset = inccmdOffset(cmdOffset,2);
      //compute new offset
}
```

The firmware transfer tools are in place. All we need to do is apply them in the correct sequence according to the *CMD_TEXT* syntax:

```
#define drawText(x,y,font,options,txt)
cmd32(CMD_TEXT);                        \
      cmd16(x);                         \
      cmd16(y);                         \
      cmd16(font);                      \
      cmd16(options);                   \
      cmdStr(txt);
```

■ Screenshot 2. Drawing a button has a lot in common with drawing text. The difference lies in the button size and the button appearance arguments.

Okay. Let's get back in the truck. The *comStr* portion of the *drawText* macro gets its text data from a buffer (*txBuf*) which is actually an array we fill with the C language *sprintf* function:

```
sprintf(txtBuf,"Design
Cycle");
//write to txtBuf
drawText(106, 77, 31,
0,txtBuf);
//make it so Number 1
```

The *"Design Cycle"* bits are at work. So, let's turn our attention to the button. We use the good old *sprintf* function to once again fill a buffer we call *btnMsg*:

```
sprintf(btnMsg,"OK");
drawButton(169, 139, 127, 55, 31, 0,btnMsg);
```

The coordinates, button size, label font, and button appearance were copied from the button widget I placed into the FT800 Editor window. **Screenshot 2** details the *CMD_BUTTON* syntax which was used in the assembly of the *drawButton* macro:

```
#define drawButton(x,y,w,h,font,options,btxt)
    cmd32(CMD_BUTTON);              \
    cmd16(x);                       \
    cmd16(y);                       \
    cmd16(w);                       \
    cmd16(h);                       \
    cmd16(font);                    \
    cmd16(options);                 \
    cmdStr(btxt);
```

At this point, we have drawn a picture that consists of text and a button. However, the Display List we have created is still behind curtains. To present our new List, we must close out the Display List code segment and instruct the FT800 to swap in the new List. Once the swap has occurred, writing the new *cmdOffset* value to *REG_CMD_WRITE* will force the FT800 to execute the commands in the ring buffer that reside between the offset held in the *REG_CMD_READ* register and the new offset we loaded into the *REG_CMD_WRITE* register. The code looks like this:

```
cmd32(DISPLAY());
      //end the Display List code
cmd32(CMD_SWAP);
      //swap the new Display List in
wr32(REG_CMD_WRITE,cmdOffset);
      //execute the new Display List
```

We're still not done. It would be nice if we actually enabled the display. To do this, we will use the FT800's GPIO bit 7:

```
wr8(REG_GPIO_DIR,(rd8(REG_GPIO_DIR)) | 0x80);
      //set the I/O bit direction as output
wr8(REG_GPIO,(rd8(REG_GPIO_DIR)) | 0x80);
      //set the bit and enable the display
wr8(REG_PCLK,5);
      //make the display objects visible
```

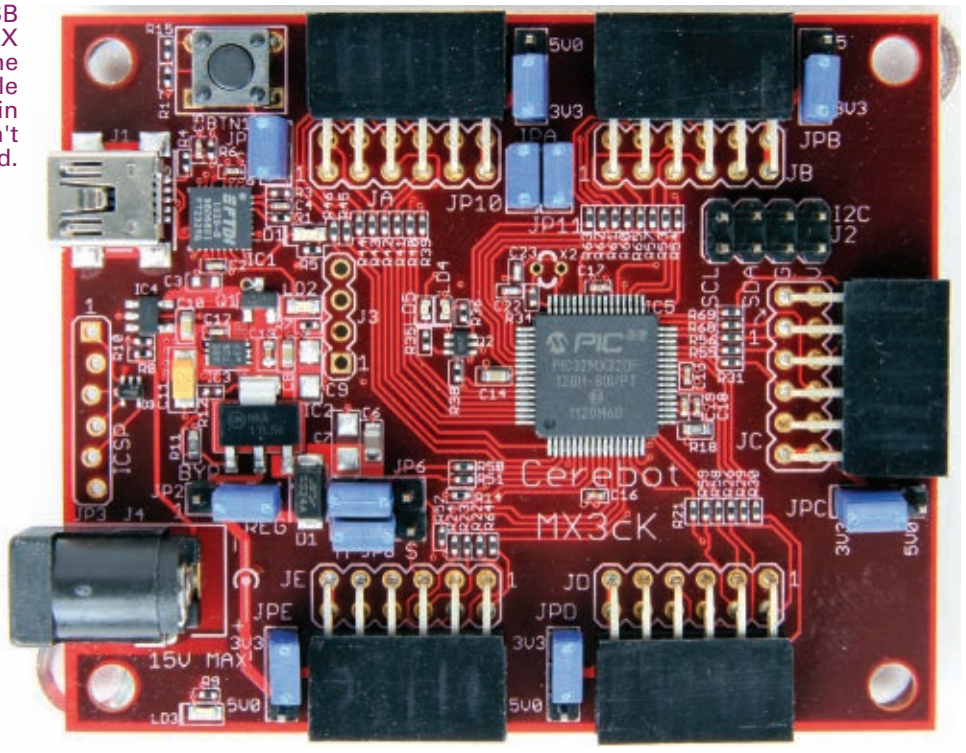
At this point, we can write and swap in Display Lists as our application requires.

Getting Touchy Feely

As it stands, touching the OK button will result in loading the default value (0xFF) into the tag buffer. The contents of the tag buffer are used to identify the widget that is being touched. The tag buffer value can range from 0x01 to 0xFF.

We have the ability to control the value that a widget reports to the tag buffer by loading the TAG register with the desired tag buffer value. To assign our OK button a tag value of 0x01, we simply insert the *load TAG* register command right before we code the button:

```
sprintf(txtBuf,"Design Cycle");
  //load txtBuf with string data
drawText(106, 77, 31, 0,txtBuf);
  //print text to LCD panel
sprintf(btnMsg,"OK");
  //load btnMsg with string data
cmd32(TAG(0x01));
  //assign tag value of 0x01 button
drawButton(169, 139, 127, 55, 31,
0,btnMsg);
  //draw the button
```



■ Photo 2. The FTDI-powered USB port is just one of the many PIC32MX peripherals that are exposed on the MX3. The MX3's 12-pin female headers allow us to easily plug in additional peripherals that aren't native to the MX3 main board.

To verify the tag value assignment, I activated the Digilent MX3's UART1 which is hardwired to an FTDI FT232RL USB-to-RS-232 converter IC. The FTDI IC and associated circuitry can be seen in the upper left corner of **Photo 2**. Rather than feed each character to UART1, it's easier to use the *printf* function. However, we have to lay the ground work first in the hardware initialize function. We must redirect the *printf* output to UART1 and enable UART1 with some C statements. Here's how that is done:

```
//************************************************
    //* UART1 REDIRECTOR (FOR PRINTF)

//************************************************
    void _mon_putc(char c)
    {
       U1TXREG = c;
       while(U1STAbits.TRMT == 0);
    }

    setbuf(stdout,NULL);
       //UART1 REDIRECT FOR USE WITH PRINTF

    //Initialize UART1
    UARTEnable(UART1, UART_DISABLE_FLAGS(UART_
    PERIPHERAL | UART_RX | UART_TX));
    INTEnable(INT_SOURCE_UART_RX(UART1),
    INT_DISABLED);
    UARTConfigure(UART1, UART_ENABLE_PINS_TX_
    RX_ONLY);
    UARTSetLineControl(UART1, UART_DATA_SIZE_8_
    BITS | UART_PARITY_NONE | UART_STOP_BITS_1);
    UARTSetDataRate(UART1, GetPeripheral
    Clock(),115200);
       // Configure UART1 RX Interrupt
    INTSetVectorPriority(INT_VECTOR_UART(UART1),
     INT_PRIORITY_LEVEL_2);
    INTSetVectorSubPriority(INT_VECTOR
    _UART(UART1), INT_SUB_PRIORITY_LEVEL_0);
    UARTEnable(UART1, UART_ENABLE_FLAGS(UART_
    PERIPHERAL | UART_RX | UART_TX));
```

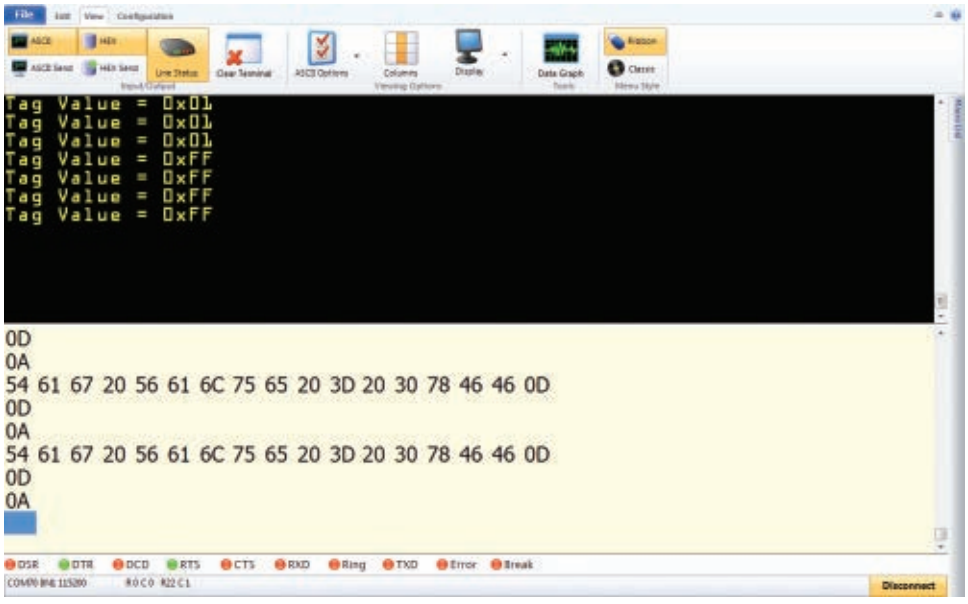
The *mon_putc* function is pretty cool. It overrides the weak *putc* function that is included in the compiler library. You can use it to redirect almost anything to *stdout*. For instance, if you want your *printf* output to go to an LCD, you simply put your LCD *write* code within the *mon_putc* braces.

Before we start pointing fingers at the LCD, it would be a good idea to give the FT800 an idea about where things are on the LCD's XY plane. We do this by executing the *calibrate* command. The calibration process places three touch dots on the screen. XY information gleaned from the touching of the calibration dots is stored as a matrix in the *REG_TOUCH_TRANSFORM_A-F* registers. The calibration routine needs to be run from within its own Display List:

```
void calTouch(void)
{
     cmd32(CMD_DLSTART);
        //start of Display List
     cmd32(CLEAR(1,1,1));
        //clear color, stencil & tag buffers
     sprintf(txtBuf,"Please tap on the dot");
        //instructional message
     drawText(144, 120, 28, 0, txtBuf);
        //write text message to LCD
     cmd32(CMD_CALIBRATE);
        //execute calibrate command
     cmd32(DISPLAY());
        //end of Display List
}
```

The LCD touch engine is ready to roll. However, the calibration Display List is currently in control. To display the OK button, we need to swap in our Design Cycle/OK button Display List that tags the OK button as 0x01:

```
    cmd32(CMD_DLSTART);
    cmd32(CLEAR_COLOR_RGB(0,0,0));
```



■ Screenshot 3. The FT800 register set allows us to get a lot deeper than just a touch and a tag value. A touch generates much more information than the average application needs to act on.

RESOURCES

Digilent, Inc.
chipKit MX3
www.digilentinc.com

FTDI
FT800
www.ftdichip.com

Microchip
XC32 C Compiler
MPLAB X
PICkit 3
www.microchip.com

```
cmd32(CLEAR(1,1,1));
sprintf(txtBuf,"Design Cycle");
drawText(106, 77, 31, 0,txtBuf);
sprintf(btnMsg,"OK");
cmd32(TAG(0x01));
drawButton(169, 139, 127, 55, 31, 0,btnMsg);
cmd32(DISPLAY());
wr32(REG_CMD_WRITE,cmdOffset);
wr32(REG_TOUCH_RZTHRESH,1200);
    //set touch screen sensitivity
wr8(REG_GPIO_DIR,(rd8(REG_GPIO_DIR))
| 0x80);
wr8(REG_GPIO,(rd8(REG_GPIO_DIR)) | 0x80);
wr8(REG_PCLK,5);
```

The Design Cycle/OK button Display List code is nothing you haven't seen before. The exception is a line of code that sets the touch screen sensitivity.

All Systems Go

Currently, we have a calibrated touch screen with the Design Cycle/OK button Display List swapped in and an active UART1. The ultimate goal is to touch the OK button, generate a tag value, read the tag value, and send the value to a serial monitor application running on a laptop.

For the purposes of our discussion, the touch sense operation is coded as an endless loop. The loop begins by reading the tag value from the *REG_TOUCH_TAG* register:

```
do{
        curTag = rd8(REG_TOUCH_TAG);
```

At this point, we may or may not have a valid tag value. Recall that the default tag register value is 0xFF unless we specify a value for each widget in our Display List. Valid tag value or not, the next step is to see if a touch has occurred:

```
btnPressed = ((rd32(REG_TOUCH_DIRECT_XY)
>> 31) & 0x01);
```

If a touch is sensed, the *btnPressed* variable value will be loaded with 0x01. The FT800 is capable of scanning the OK button thousands of times before we can lift our finger from the touch screen. So, we will use some caveman DSP filtering to limit the sample quantity to 50 scans:

```
if((btnPressed == 1) && (prevTag != 0))
{
        btnIndx++;
        if(++btnIndx > 50)
        {
                btnIndx = 0;
        }

}
btnBuf[btnIndx] = curTag;
```

Although the PIC32MX mounted on the MX3 can easily handle the continuous serial touch traffic, there is absolutely no need for us to send invalid tag buffer values (0x00). The *printf* I/O we established earlier makes is easy to send both descriptive ASCII text and the actual hexadecimal tag value that was sensed during the touch event:

```
        if(btnBuf[btnIndx] != 0x00)
        {
                while(cmdInQueue());
                printf("Tag Value =
                0x%02X\r\n",btnBuf[btnIndx]);
        }
        delayms(100);
        prevTag = curTag;
}while(1);
```

Screenshot 3 tells the story behind the *printf*-based code we just wrote. The *prevTag* variable eliminates running through our caveman DSP code needlessly.

HP-65

I thought I was the boss when I made my HP-65 calculator jump through my (at the time) clever display-based programs. I recall my mom getting mad at me for locking myself in my room and then writing HP-65 "games" which were based solely on logical mathematical inputs and visual mathematical outputs. The visuals were based on numerical results generated by keyboard stimulus. The HP-65 display consisted of a bunch of tiny seven-segment LEDs. The real fun was storing my "programs" on magnetic cards. The HP-65 was equipped with a magnetic card reader/writer. That calculator did everything but talk.

Those were the days. As it relates to the FT800, you not only have the tools to display graphic widgets and text, you can use those same basic FT800 read/write functions that we wrote over this series of FT800 discussions to generate sound:

```
wr8(REG_VOL_SOUND,0xFF);
    //set the volume to maximum
wr16(REG_SOUND, (0x6C<< 8) | 0x41);
    // C8 MIDI note on xylophone
wr8(REG_PLAY, 1);
    // play the sound
```

I've provided the base you need to get started with the FT800. It's up to you to make some noise. **NV**

NEW PRODUCTS

- HARDWARE
- SOFTWARE
- GADGETS
- TOOLS

PINION GEARS

In addition to their newest Actobotics™ hub, servo mount, and plain bore gears, ServoCity now boasts a large selection of 32 and 48 pitch pinion gears. Offering both metric (3 mm and 6 mm) and SAE (1/8" and 1/4") bore diameters, these gears will work for all kinds of applications.

Each bore diameter comes in a variety of sizes, ranging from 12 teeth up to 36 teeth. Simply slide the gear on the shaft and tighten the setscrew. Manufactured from hardened brass gear stock for extreme durability and strength, these new pinion gears start at US$7.99 each.

PLANETARY GEARMOTORS

Adding to their extensive line of gearmotors, ServoCity recently released two new styles of Actobotics planetary gearmotors: 3V-12V precision planetary gearmotors and 6V-12V heavy duty precision planetary gearmotors. Both styles offer all-metal gears and dual ball bearings to ensure the motor will hold up in even the harshest applications. The smaller 3V-12V gearmotors have a 3 mm shaft, while the larger 6V-12V gearmotors have a 6 mm shaft.

ServoCity has many compatible accessories — including pinion gears, hubs, and couplers — that attach directly to the shaft of the gearmotor, as well as several compatible motor mounts. With 24 RPMs to choose from, there's the perfect drive motor for a robot, camera system, or other custom project. Gearmotors start at US$27.99.

For more information, contact:
ServoCity
Web: **www.servocity.com**

LED BARREL CONNECTOR LIGHT

The J2 LED Lighting, LLC MEL (Micro Effects Light) is a series of simple low cost LED light sources with a built-in DC barrel type connector. The 2.1 mm x 5.5 mm barrel connector is very common for many low voltage DC power applications, simplifying the building of a micro effects lighting system in a modular fashion.

The MEL operates from 9–12 volts DC which makes a simple effects system operating from a nine volt battery possible. More complex systems with numerous micro effects lights are possible from a 12 volt DC power source.

Lighting systems can range from a single MEL to over 100 MELs, depending on the complexity of

the lighting project. The MEL is optimized for high light output at nine volts DC, with a nominal power of 0.18 watts operating at 20 mA ref. With an input of 12 volts, a power consumption of 0.34 watts is typical operating at 28 mA ref.

With input higher than 12 volts, there is a diminishing light output due to thermal loading of the LED. The MEL is dimmable by either PWM (Pulse Width Modulation) type dimmers or with variable voltage input from 7–12 volts DC. The MEL can be made with pure fully saturated LED colors and with phosphor converted colors for a wide color gamut. The LED lens is water clear with a 30 degree output angle for a narrow beam.

The MEL can be operated from a nine volt battery for portable applications. When using a nine volt battery, it is recommended that the number of MELs does not exceed six in order to limit the load current to about 120 mA for battery life. The MEL's equipped with a barrel connector type DC power jack can mate to a plug of the same type, and it uses a positive center power connection.

The MEL is compatible with other J2 LED Lighting accessories of the same connector type which include two- and three-way DC barrel connector splitter cable harnesses, pigtail harnesses, MED1 micro effects dimmers, nine volt DC barrel battery clips, and DC barrel-to-wire terminal block adapters.

The axial body of the MEL LED is of a small size and is molded in black ABS resin. The MEL body overall length is 1.42" (36 mm) and the diameter is 0.48" (12.2 mm) nominal. The MEL is well suited for various LED lighting applications such as:

- Scenery/props/models/animatronics/robotics
- FX special effects/costumes
- Theatrical stage and set obstruction marking
- Theming, décor lighting, and promotion display products
- Amusement, arcade, gaming, and entertainment lighting
- Specialty miniature accent lighting, micro fixtures, and micro ceiling drop pendant lighting
- Sales and trade show displays and signage

Pricing is the same regardless of color:
1) 1-10 pcs = US$1.99 each
2) 11-25 pcs = US$1.84 each
3) 25-100 pcs = US$1.72 each

For more information, contact:
J2 LED Lighting
Web: **www.j2ledlighting.com**

PCB PRICE CALCULATOR

PCBShopper.com is a new website designed for electronics

hobbyists and professionals. It helps find the cheapest source of printed circuit boards for projects in prototype quantities like five or 10, or small batch quantities up to 1,000. List the board's size, number of layers, quantity, and the preferred solder mask color, and in seconds the price calculator will show prices and delivery times from many different PCB manufacturers. Currently, there are 16 manufacturers, with more being added all the time. To find the lowest rates, there's a sort by price feature; to find the fastest, sort the results by delivery time.

PCBShopper.com also maintains a list of free CAD programs. Since many of them are feature-limited versions of expensive commercial software, PCBShopper details what the imposed limitations are.

For more information, contact:
PCBShopper
Web: **www.pcbshopper.com**

MIXED SIGNAL TEST, MEASUREMENT, AND DATA ACQUISITION SYSTEM

BitScope is now offering a unique low cost mixed signal test, measurement, and data acquisition system for embedded computing. Called the Bitscope Micro, it's a tiny low power USB connected device with comprehensive cross-platform software and libraries. Bitscope Micro can be the perfect companion for embedded systems such as Raspberry Pi, and it supports remote and shared access via TCP/IP networks. Use it to build custom data acquisition, telemetry, or closed-loop test systems, or simply as a low cost "go anywhere" problem solver that can fit in the palm of your hand.

The BitScope Micro is a mixed signal scope in a probe. Features include:

- 20 MHz bandwidth
- 40 MSps logic capture
- Two analog scope channels
- Two analog comparator channels
- Six logic/protocol analyzer channels
- Eight and 12 native analog sample resolution
- Decodes serial, SPI, I²C, CAN, and more
- Windows, Linux, Mac OS X, and Raspberry Pi
- Built-in analog waveform and clock generators
- User programmable, C/C++, Python, and VM API
- Tiny, light weight (12g), and water resistant

Pricing is from US$95 for 10+ (aimed at students and bulk buy), with a 1+ retail of US$145.

For more information, contact:
BitScope
Web: **www.bitscope.com**

EIGHT CHANNEL, 12-BIT, 1 GHz BANDWIDTH OSCILLOSCOPE

Teledyne LeCroy has announced the HDO8000 oscilloscope product line with eight analog input channels, 12 bits of vertical resolution utilizing their HD4096 technology, and up to 1 GHz of bandwidth. The HDO8000 oscilloscopes have maximum performance, with a wide variety of mixed signal, serial data, long memory, and probe options. Several accessories are available for the HDO8000 oscilloscopes.

The HDO8000 is ideal for high power three-phase power electronics system analysis. The global market for power electronics is growing quickly, especially in high power and three-phase energy conversion applications focused on distributed power generation (solar PV, wind, etc.) and hybrid electric and electric (HEV and EV) vehicle propulsion systems. New and expanded efficiency requirements for electric motors and variable frequency (motor) drives are also creating higher demand for three-phase power electronics.

Additionally, eight channel high definition oscilloscopes are highly useful in debugging deeply embedded systems in applications such as automotive electronic control units (ECUs), consumer appliances (e.g., washing machines, refrigerators), and industrial systems (e.g., robotics) that contain a

complex mix of power electronic, power, clock, digital logic, serial data, and analog sensor signals. More channels and more resolution provide faster insight into embedded system behavior.

The new Q-Scape™ multi-tabbed displays available provide four times the display area and better organization of large numbers of channel, zoom, and math waveforms (up to 40 total) on the oscilloscope's 12.1" WXGA high resolution display. Four tabbed displays are provided. Waveforms can simply be dragged and dropped to the desired location to conveniently organize the many different acquired and calculated waveforms for more

Continued on page 48

By Robert Reed

FREQUENCY COUNTERS AND RETROFITTING

Frequency counters have always fascinated me. Even a mediocre model has higher accuracy than any other instrument on a test bench. My first such counter was a Lampkin micrometer frequency meter which was popular in the mid-1960s. This well-designed instrument was based on a narrow range variable frequency oscillator (VFO) that was tuned with capacitor plates.

The VFO was attached to a modified machinist's micrometer with a calibrated dial plate. After a 15 minute warm-up, the VFO was zero beat to one specific frequency against a very stable compensated crystal oscillator. From that point, each frequency being tested was checked using a lookup table after zero beating the Fo under test to the calibrated dial reading.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/august2014_Reed.**

According to its specifications, the Lampkin's accuracy was only ±50 Hz. Even so, it performed well enough for my ship-to-shore radio business at the time. My frequency meter's accuracy was far better than the marine radios of that era. They fell in the medium frequency spectrum and rarely had more than six or seven crystal-controlled channels with a Federal Communications Commission (FCC) tolerance of ±100 PPM (parts per million) of carrier.

My meter served me well for years. However, the shift over to VHF gave rise to tighter frequency tolerances and meters with 80 channel capability. I knew it was time to shell out big bucks to replace the old Lampkin with a modern counter with (gasp!) a digital display. In retrospect, as marvelous as the Lampkin's engineering was, its operation almost seems comical to me now.

Frequency Counter Overview

The frequency counters currently on the market offer both high accuracy and ease of use. Even though the circuit actions are complex, the circuit itself is fairly simple. This straightforward circuit design is the result of its minimal analog circuitry, as well as the advent of VLSI (Very Large Scale Integration) logic ICs. These days, a counter whose design is not PIC based tends to follow a very similar design pattern. **Figure 1** shows a simplified block diagram of a typical counter.

The frequency being measured is labeled here as "clock." The input to block #1 uses a probe with the amplification and prescaling necessary for the counting circuits. Most logic counters are limited to 10 MHz or fewer. If the clock has sufficient amplitude, block #1 can be clocked directly into block #2. However, when frequencies exceed the 10 MHz limit, a prescaler is required to divide the clock signal until its frequency reaches a level that the counting circuits can handle.

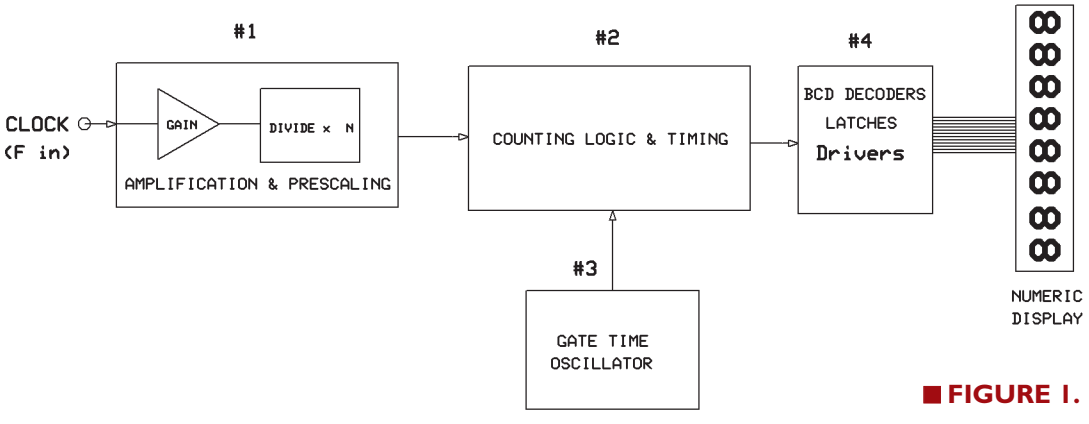
Prescaling reduces the counter's resolution, typically by an insignificant amount. When the clock's pulses enter block #2, the counter begins counting every pulse it "sees;" block #3 (the gate time oscillator) determines what the counter sees. The function of block #3 is to produce an accurate time base to gate the incoming clock signal in block #2. This time base is derived from a stable crystal oscillator and divided to a gate time that is typically either a 0.01, 0.1, 1, or 10 second period.

When this gate opens, clock pulses are fed to the counter and it starts counting. When the gate closes, the feed path is blocked, the counter stops, and it holds the last count. For example, when the gate time is 1.0 seconds and the clock rate is 1.0 kHz, exactly 1,000 pulses are counted and held in binary-coded decimal (BCD) format. This requires a four-digit display and four BCD outputs from the counter — one for each digit.

The timing circuits in block #2 ensure that all the requisite operations occur consistently and sequentially. The clock pulses are counted until the gate closes, at which time block #4 is activated. The held count is latched into block #4 and decoded BCD into seven-segment out which drives the segments of that particular digit. Every digit has its own latch, decoder, and driver, and this transfer takes place simultaneously for all digits. Immediately after the latches capture the BCD info, a reset pulse is generated and all counters are reset to zero. Shortly afterwards, the time base gate opens and the process repeats.

Here's a summary of the basic function counter operation:

1. The time base gate opens.
2. Counting begins.
3. The time base gate closes.
4. Counting stops.
5. Counter info is latched and decoded to digit segments.



■ **FIGURE 1.**

6. Counter resets to zero.
7. The time base gate opens, and the process starts all over again.

A multiplexed display has a similar operation but it is, of course, more complex. In a multiplexed configuration, a low frequency oscillator cycles through the digits and segments continuously, enabling the correct digit and its corresponding segment driver at the same time. The multiplex rate can range anywhere from 100 Hz to 500 Hz per second. This greatly reduces wiring, construction, and overall current drain because only one digit is lit at any point in time.

The frequency counter's overall accuracy is largely determined by block #3 — the time base generator. Any error that occurs is measured in PPM, and it is transferred directly to the displayed readout. The counting and timing circuits are fairly reliable. They have little bearing on overall accuracy, with the exception of the ± one count that occurs in any digital readout. This phenomenon is not a fault in the circuitry but, rather, an occurrence caused by the incoming clock signal not being phase-locked to the time base gate.

If the gate opens somewhere in the mid-point of a high clock signal, it will also close during a mid-point. This will cause the readout to register one more count than the actual clock frequency. Similarly, the readout can lose a count when the gate opens somewhere on a low clock signal. Unfortunately, there is no easy way to avoid this situation. However, since it only affects the LSD (least significant digit), more digits reduce the error.

This article focuses on the construction of frequency counters ranging from a simple retrofit unit to a full-blown universal counter with all the bells and whistles. All of these counters are based on a marvelous chip developed in the 1980s: the Intersil ICM7216 series. I first became aware of this chip 15-20 years ago. At that time, it was no longer in production. These chips were still available but only with a $500 minimum order from secondary suppliers. In spite of these factors, I kept my eye on these chips over the years waiting for them to be affordable, but to no avail.

Frequency counters I built in the interim were "stick built" from multiple single-function logic chips. Each counter required many chips and much wiring to construct. All the while, I was chomping at the bit for those hard-to-find Intersil chips. A few years ago, small time distributors started listing a plethora of electronic ICs from China. Not only were these chips (and many others) now available, they could be ordered in small quantities — even individually — for an affordable price.

Each of the four varieties of the ICM7216 has its own suffix: A, B, C, and D. There are minor differences between the chips, and one will be more suited for different applications than the other. While they all make suitable frequency counters, the A and B varieties have options that enable them to make complete universal counters that can measure period, pulse width, frequency ratio, and totalizing. The A and C versions are designed for common anode displays, while the B and D varieties are for common cathode displays.

These counter chips require very little support circuitry and make construction so much easier than their predecessors. The basic frequency counter only needs a few components in addition to the chip. Adding a few switches enables them to perform practically any function you might need from a universal counter. With such enhancements as an amplifier/prescaler and an external TCXO time base oscillator, your counter can rival most high-end models on the market with an upper frequency limit of 800 MHz or higher, and accuracy exceeding one PPM!

In the June 2014 issue of *Nuts & Volts*, I wrote an article about installing a counter in a 150 MHz RF signal generator. This installation provides an excellent example of retrofitting a frequency counter. When retrofitting frequency counters, the first thing to consider is matching the counter design and the piece of equipment in which you will install it. Factors that come into play include the number of display digits, the accuracy and resolution required, and the prescaling (if needed). For the RF signal generator, I needed at least three display digits because I would be counting in MHz.

Prescaling was necessary since I would be measuring frequencies above the 10 MHz limit of the counter chip. The generator's specs list the limit of its short-term stability as 100 PPM. However, according to my own testing, the stability across its range averaged closer to 20 PPM. That limit equates to either 200 Hz at 10 MHz or 2,000 Hz at 100 MHz.

I wanted to resolve somewhat beyond 2,000 Hz to more accurately measure the average frequency outputs of this generator. I arbitrarily determined that a 1 kHz resolution would be optimal stability across its total bandwidth (0.300-150.000 MHz). It would be pointless to have a resolution of 1 Hz or 10 Hz if the generator was not stable enough to sit in either spot for more than a few seconds.

I determined that I needed to cover a range of 150 MHz with a resolution of 1 kHz which requires six display digits (e.g. xxx.xxx MHz). Since the counter's input frequency limit is 10 MHz, I needed to reduce that 150 MHz input to below 10 MHz. This required a prescaling division of 100 so that the highest frequency the counter would see would be 1.5 MHz.

I recommend prescaling in multiples of ten; doing so simplifies the gate period and timing circuitry. Counter's digital readouts rarely count the actual frequency displayed. By prescaling in multiples of ten and repositioning the decimal point, the numbers come up

correctly (e.g., 150.000 MHz into prescaler, 1.50000 MHz into counter, move the decimal two places to the right, and label readout in MHz; final result 150.000 MHz).

A frequency counter's accuracy and resolution requirements are closely tied. In this particular case, I had a resolution of 1 kHz which equated to 100 PPM (and more when measuring any frequency below 10 MHz), which meant I couldn't discern any error less than this amount. At measurements of 100 MHz, the least discernable error would be 10 PPM. Any accuracy of the counter less than 10 PPM would not even register; due to the stated resolution of 1 kHz, it would need one more digit of resolution to display that error.

The crystal I chose for this application has an overall temperature coefficient of ±30 PPM across its whole temperature range. With the added trimmer cap, it can be set very accurately. Since its primary usage will be in a lab environment, it will probably hold within ±5 PPM in a lab, and that will suffice for this application.

The final factor that I needed to address in this design was the range/counter gate time period. A 1.0 second gate time gives a nice steady display, but is too slow for tuning to any given frequency. A 0.1 second gate is great for tuning since it tracks the frequency as fast as the tuning control can be turned. However, once it reaches the frequency limit, the counter might emit an annoying flicker on the LSD digit. I found the ideal operation has a refresh rate of 0.3 seconds. This rate occurs with a 0.1 second gate time and the chip's built-in 0.2 second 'between reading' time, thus giving the desired refresh rate of about three times a second.

If I were designing this counter for a higher caliber generator — that is, an extremely accurate and stable one with a 1 GHz tuning range — I would probably end up with eight digits and a resolution of 100 Hz, plus an accuracy of less than 1 PPM. On the other hand, if this generator were to be used as an internal combustion engine tachometer, a four-digit display, no prescaling, 1 Hz resolution with an allowed accuracy of 500 PPM, and a gate time of 1.0 seconds would certainly suffice. A key

PARTS LIST

ITEM	DESCRIPTION	PART #	SOURCE	COST
All resistors are 1/4 watt 5% carbon film.				
RESISTORS				
R1*	680			
R2	22 meg			
R3	100			
R4	100K			
R5	10K			
R6	10K			
RP1*	680 x 8			
CAPACITORS				
C1, C3, C5, C8	0.1 µF			
C2	0.001 µF			
C4, C9	0.01 µF			
C6	10-50 pF variable			
C7	36 pF			
C10	470 µF			
CRYSTAL				
10.0000 MHz		TWX 887-1235-ND	Digi-Key	$0.50
FERRITE BEAD				
FB-1	J. W. Miller	F43-287-RC	Mouser	$0.20
INTEGRATED CIRCUITS				
IC1		11C90	eBay	$5
IC2		74HC390	Mouser	$1
IC3		ICM7216D	eBay	$7
DISPLAY				
Two-digit	LITEON	LTD-4708JR	Mouser	$2
Four-digit	LITEON	LTC-4727JR	Mouser	$3
CIRCUIT BOARD				
RS-276-168			RadioShack	$3.50

* Depends on display drive current desired.

Datasheets for all ICs are readily available on the Internet.
For email packet or any other assistance required, send a note to rjr@ncweb.com. Keep in mind, listed prices are subject to change.

factor in this process is matching the counter specifications to the task at hand.

The Retrofit Counter

The following is the counter I designed based on the ICM7216D chip and built for my previous articles ("180 MHz Sweep Generator" in the December 2013 issue and the more recent "150 MHz RF Signal Generator"). These six-digit readouts can be easily modified to suit your application; the 7216 chip can handle eight display digits of readout if necessary.

This chip made life so easy, I kept checking and rechecking to see if I missed something as construction went so much faster on this project. It was constructed on a RadioShack RS-276-168B circuit board, and it was as if the board was made for this project. Everything just fell into place with room to spare. I have always liked these boards as they are padded for standard DIP sockets with the power supply rails running lengthwise directly under the IC chip socket.

Due to the 7216 wide 28 DIP configuration, some modification was required. **Figure 3** shows the completed board top side and the unmodified board foil side. One trace has to be cut and several others jumpered on the power supply rails. Also, there is a small amount of foil grinding to do where the ICM7216D is located. I will be omitting a lot of details from here on out due to size restrictions for publication, but as I have done in past articles, there will be an email packet with additional pictures and construction information I can provide on request for those who are interested. You can also check for updates at the article link.

Once you modify the circuit board, you can mount the IC sockets (I used sockets for each chip). The pins on the right side of the 7216 socket will end up in the ground-out foil area; they require a series of short jumpers from the pins to the respective solder pads adjoining the socket. You might not need the resistor pack RP1 socket for your LED display. However, it is so cheap and when dropped in place mates up perfectly with the segment outputs. In fact, it drops onto the same solder pads as the 7216 so there are not even any connecting wires needed.

The LED displays that appear in the Parts List are high-efficiency red. These displays emit a lot of light using very little current drive; the resistor pack was necessary to limit their current draw. Using an assortment of resistor packs, it was easy to determine the degree of brightness I wanted just by swapping them out. The value I chose was 680 ohms; I could have increased this to 1K ohm and achieved sufficient brightness. Even if your display needed more drive than that, it's still a good idea to have some limiting resistance, if for no other reason than to help extend the life of the 7216.

The resistor packs contain eight resistors and require a 16 DIP socket. Seven of these resistors are used in a series with the seven-segment outputs, but the one in the middle just happens to line up where the 7216 V+ pin is located. This resistor was left unterminated on the other end. You could do a lot of grinding in this area and free it up for the

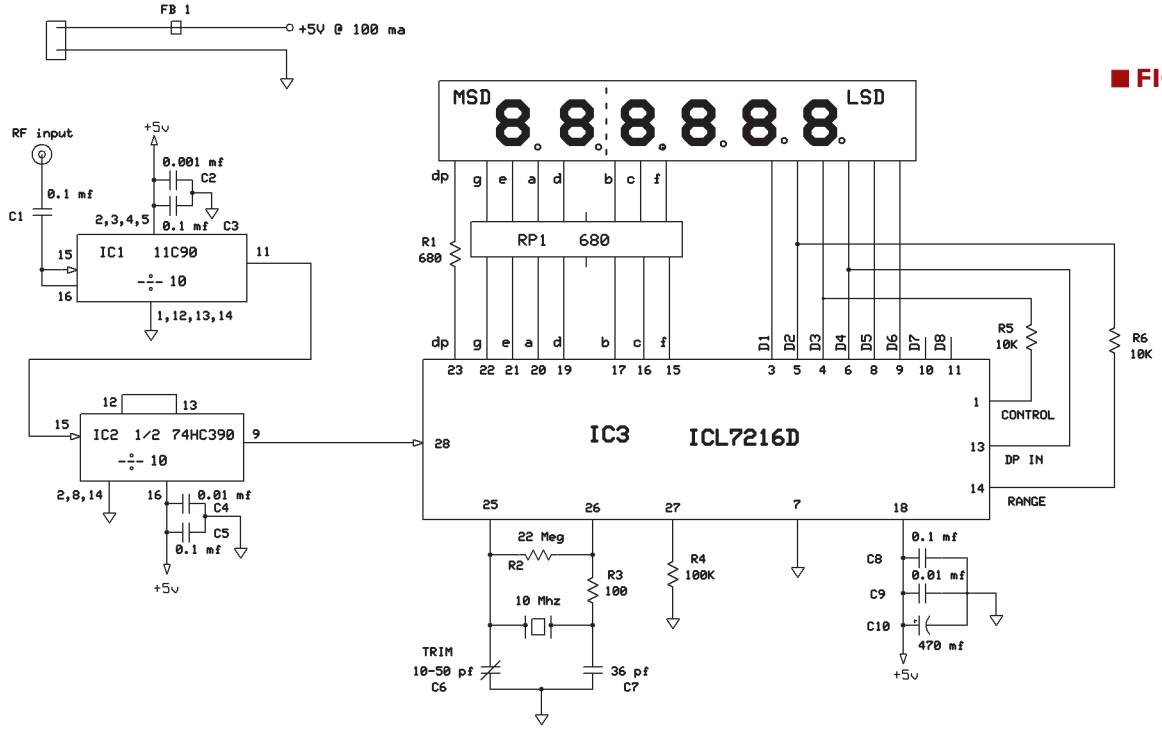
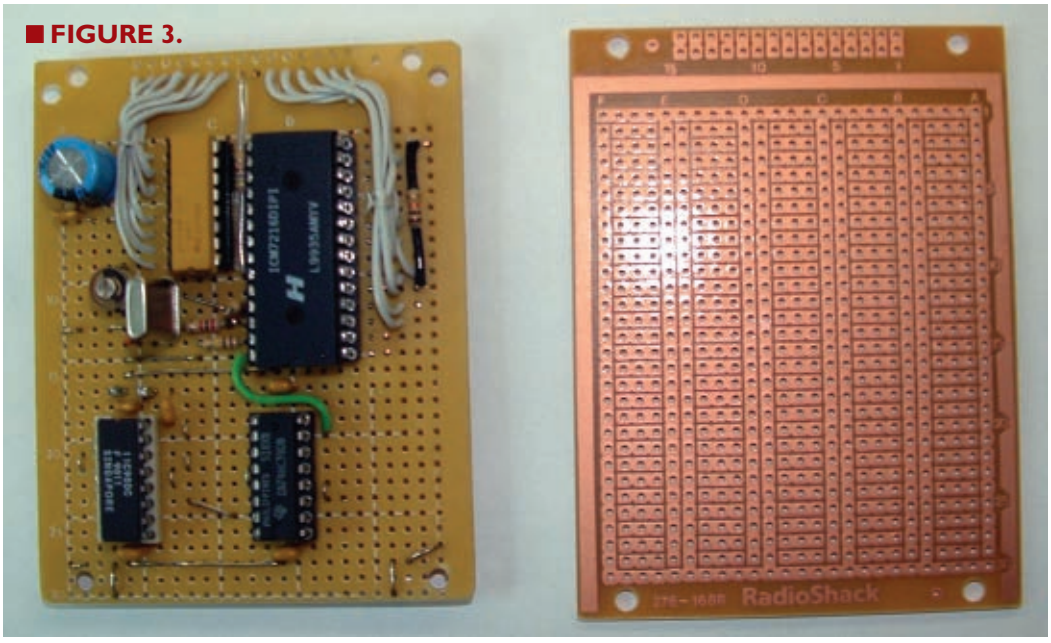


FIGURE 2.

decimal output segment, but I found it easier to just run that output directly to the display through its own 680 ohm resistor.

All of the 7216 digit and segment lines were run to the front of the board and terminated at the inboard row of holes. When the display assembly is complete and wired, those leads will be connected to the mating outboard row of holes. This configuration makes assembly simple and straightforward. The 7216 series of chips can perform a lot of functions because the multiplexed digit output pulses it uses act as enabling triggers for various purposes.

■ **FIGURE 3.**



We will use two of these functions in this design: the range (gate period) and external decimal point control. To enable the external decimal point feature, you must make a connection from the control pin (1) to the D3 output pin (4) through a 10K resistor. My displayed decimal point is on LED digit 4; I had to run a wire from the decimal-in pin (13) to D4 (6) on the chip. I selected a 0.1 second gate time; activating this function required a connection from range (14) to D2 (5) through a 10K resistor. The 10K resistors suppress stray pickup on these lines. A 100K resistor to ground from pin 27 disables the hold function. The only remaining step in the frequency counter retrofit process is to wire the oscillator.

Connect the oscillator as shown. The trim cap C6 can be made up of any combination of fixed and variable capacitance, as long as you end up with about 30-35 pF total with the trimmer at mid-range. (Note: The crystal shown in **Figure 3** is not the one listed in the Parts List. It was replaced after the photo shoot was completed. The replacement has better frequency stability and aging characteristics than the original.)

All crystal resonant frequencies will change somewhat due to aging. The order of acceleration of this phenomenon is as follows:

1. Unavoidable chemical impurities from production, even when just sitting on the shelf; very minute.
2. Active operation; this one ages at 3 PPM per year — a better than average rate.
3. High ambient temperature or over-driving, maybe 10-15 PPM per year.

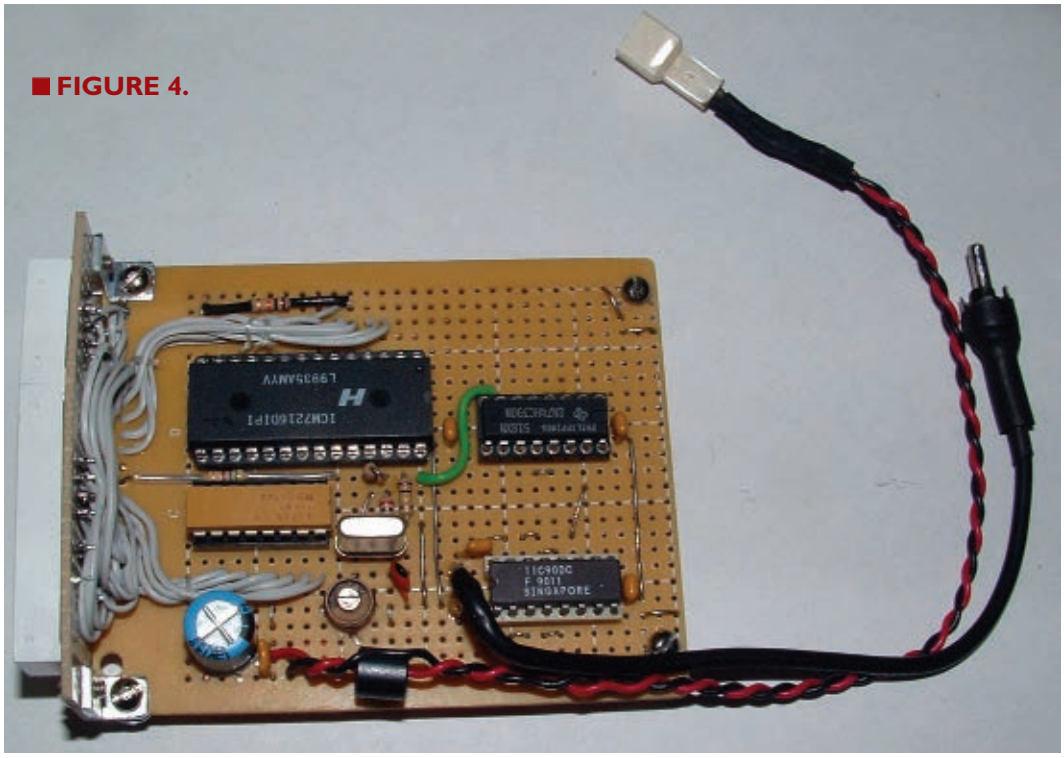
Typically, after the first year of operation, the aging rate slows down significantly. The added trimmer makes quick work of putting it back on frequency when required. As for the three bypass capacitors coming off of V+ (pin 18), they were placed as follows: 0.01 µF at the pin 1 end of the chip; 0.1 µF at the pin 14 end of the chip; and 470 µF at the incoming V+ connection point. These drop in conveniently right over the power supply rails in each location.

The prescaler is made up of two chips: an 11C90 and a 74HC390. The 11C90 is the first stage in line to receive the clock signal (F in) and is configured to divide by 10. This chip is a member of the ECL (Emitter Coupled Logic) family. The key to its high speed operation is that it never enters into saturation or cutoff in its logic level swings, which are approximately 800 mVpp. Operating in this mode, PN junctions can perform extremely fast switching, and this particular one is rated at over 800 MHz.

The input is capacitively coupled so that it can be tied to an internal DC bias supply. This sets its operating point to its most sensitive region and increases its sensitivity to approximately 65 mV rms. This chip also has an ECL-to-TTL translator built into its output, so it will have no problem driving succeeding TTL or CMOS chips that follow it.

Of particular interest is its ability to operate on incoming signals at low frequencies and with slow rise times. Most ECL chips do not operate properly below 10 MHz due to the slower rise times incurred. The 74HC390 is a dual bi-quinary divider, and only half the input is used and configured as an x10 divider. The clock pulse from its output drives the input to the 7216 (pin 28) and is exactly the input frequency divided by 100.

The circuitry for the entire prescaler is so straightforward and simple, it requires no further



■ FIGURE 4.

information. For the most part, it does not require any components, and it only needs point-to-point wiring. However, its construction requires further discussion.

Each stage needs very short leads and a good ground system around it for good individual stage neutralization. This is where the RS circuit board shines with its power supply rails running directly under the chip as it facilitates tight wiring and bypassing.

Place the bypass capacitors for each stage (refer back to **Figure 2**) as follows: 11C90 - C2 at the input end of chip (15), C3 at the output end of chip (11); 74HC390 - C4 at the input end of chip (15), C5 at the output end of chip (9). As previously mentioned, these caps drop in conveniently right over the power rails.

Note: The 74HC390 shows its output taken off pin 7 (see **Figure 3**); this output should actually come off pin 9 as shown in the schematic (again, this change was made after the photo shoot). I decided to use the other half of this divider to keep connecting leads shorter. I used RG-174 cable for the clock lead (Fin) and attached it right at the capacitive coupled input to the 11C90. The other end is terminated with a standard RCA phono plug. The incoming power leads attach directly to C10.

I cut a piece of perfboard to the required size and super glued the LED digits to it. Then, I attached some small L brackets to it so that I could install it to the circuit board later.

Wire up the displays and leave two inch leads extending from it. Lay the RS board face up and the display board face down in front of it. Connect the leads to their proper locations on the RS board; then, fold it up and attach it to the RS board using the pre-installed brackets. The screws used here must be long enough to run completely through the brackets, RS board, spacers, and metal housing which you will then secure with nuts. Secure the opposite end of the board in a similar way. Obviously, this will require some prior machining. (The e-packet will contain all the details.) The completed circuit/display board is shown in **Figure 4**. The housing for it measures 3" wide x 3-7/8" deep x 1" high, and it's shown in **Figure 5**. At this point, the counter is retrofitted and is ready for testing and final installation.

Closing Notes

The retrofit just described was a joy to build because it was so simple! I know some readers might say this could be done just as easily with a microcontroller, but I'm not so sure I would agree. Having seen many designs on the Internet, I would not argue that it looks simple, but I see a lot missing such as prescalers, function controls, and actual display circuits in lieu of a laptop.

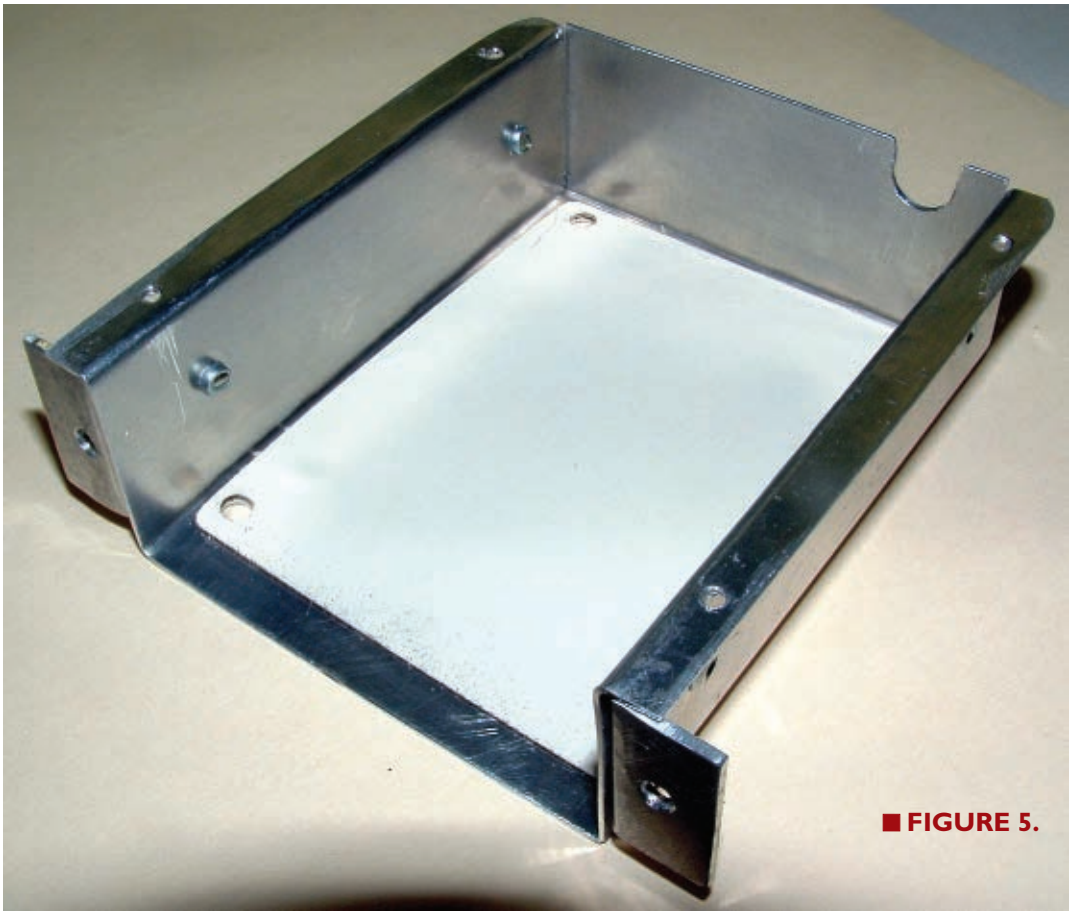
As one who despises pages of software and loves through-hole construction, this chip is a winner for me. The support circuitry needed is almost next to nothing, and the features that can be added are effortless due to the simple built-in connections to achieve such. This chip has full capability for eight-digit displays (either LED or LCD), numerous modes of operation, leading zero blanking, display testing, and the list goes on.

As a member of the CMOS family of chips, its current draw is negligible. The design just covered requires 100 mA max current. Breaking it down, that's 14 mA for the display; 2 mA for the crystal oscillator; 1 or 2 mA for the chip itself; and a whopping 83 mA for the prescaler (the price you pay for wide band/high speed operation).

If you opt to build the retrofit but can't 'shoe horn' it into existing equipment, you could put the 5V regulator chip on board and power it from a wall transformer. Devise some quick disconnect mount for attachment;

even Velcro™ would work. Want to take it to the max (e.g., a stand-alone universal counter)? Use the full eight-digit capability and either the A or B version (depending on the common connection of the display digits), all the bells and whistles, and an internal power supply. In either case, the supplies will need +5V at 100 mA or more, depending on how many extra features you want.

The Tamura transformer 3FS-316 and a 7805 regulator would work perfectly here. It is small, affordable, and available through the usual distributors. For use in low frequency operation with an upper limit of 10 MHz, you can bypass the prescaler and go direct; the resolution can be set for 1 Hz.



■ FIGURE 5.

For counting operations upwards of 1 GHz (probably closer to 800 MHz) with an eight-segment display, the resolution will be 100 Hz, still using the divide by 100 prescaler. I tested the prescaler shown in **Figure 2** across a wide range of frequencies, and it performed flawlessly to 500 MHz with even sensitivity across that range (65 mV rms). However, at 525 MHz, it started to sputter. This was due to the 74HC390 reaching its upper frequency limit of 50+ MHz. No problem for the described retro counter, but if you need to push it higher a modification is needed.

The insertion of a 74VHC74 bistable between the 11C90 and the 74HC390 will increase the prescaler capability to 1 GHz. This chip is of the 74VHC family — a very high speed dual D type flip-flop; only one side is used. It will handle well above 150 MHz clock signals, and by connecting this in 'toggle' mode, operation will cut the clock signal in half at this point. This, in turn, will supply the 74HC390 with a lower speed that it can handle with ease. The 74HC390 will have to be reconfigured for quinary operation (divide by 5), which is a very easy task; this will preserve the overall divide by 100 of the prescaler.

I would recommend a preamp on the input of maybe 10X gain, and that the entire prescaler be constructed on single-sided copper board due to its higher frequency range. Of course, there are myriad prescalers out there that could be used, but a lot of them have binary division ratios which will require some added circuitry to end up with the correct division ratio.

If you decide to build up this counter, make sure you download the datasheets for all the chips before you even start. The 7216 has all suffix versions on the same datasheet. Read through this at least once, if for no other reason than just to get a better understanding of what you are working with. To me, the datasheets seemed a bit sketchy and had a few typos. I actually had to read it twice before I had an adequate understanding of it, but all in all gained very good info. The other two datasheets are pretty straightforward. Also worth mentioning here is that the Intersil 72xx series has many related counter chips that have options for digital programing, etc., that might be of interest to you.

I hope I have covered all the bases here without any holes in my explanation, but the e-packet will fill in any missing gaps. One thing I do know for sure — and because of the utter simplicity of these chips — I will probably be adding a lot of counters to existing equipment in the coming year. **NV**

By Frank Muratore

THE VERSATILE WIRELESS DOORBELL

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com /index.php?/magazine/article/ august2014_Muratore.**

While browsing at a bargain store, I came across a very inexpensive wireless doorbell. Although I didn't need this particular device, I couldn't pass up such a bargain (they were less than five bucks each). When it comes to wholesale electronics, I can't "resist" ... so, I bought several.

One thing my experience with electrical circuits has taught me: Advances in technology might have decreased their price points, but not their usefulness. A product with circuitry will always have a use. It might just require some repurposing.

This question piqued my interest: What practical use could I find for this device? As it turned out, I was able to evolve these low cost electrical circuits into devices that most homeowners would find invaluable.

Here are some applications to consider for repurposing the doorbell circuit:

Remote Flood Alarm: I bridged the pushbutton on the wireless doorbell's remote unit with a straightforward sensor. I fashioned it using a piece of PVC pipe with two screws protruding from it (see **Figure 1**). The circuit was sensitive enough to set off the remote chime when I poured liquid on the sensor. After one simple modification, I had a nifty remote flood alarm.

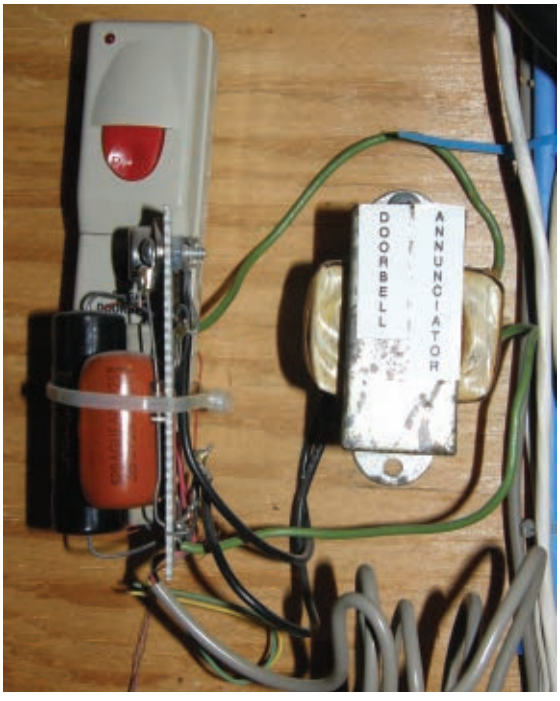
Sump Pump Failure Alarm: I have a sump pump in my basement. I fabricated another sensor which I placed slightly above the waterline (where the pump turns on). Another simple modification led to another handy notification system.

Enhanced Remote Doorbell Signal: Using a transformer and transistor (see **Figure 2**), I was able to improve the circuit's sensitivity. First, I remotely sampled a small current and triggered my doorbell. Then, I installed the wireless doorbell circuit in my existing doorbell circuit (see **Figures 3** and **4**). Once I located the bell transformer, I placed my doorbell's circuit in series with one lead going to the bell.

I relocated the remote to my backyard, so now whenever anyone rings my doorbell, I can hear the signal up to 100 feet away! This has proven invaluable when I'm gardening, grilling, or doing any other activity in my backyard.

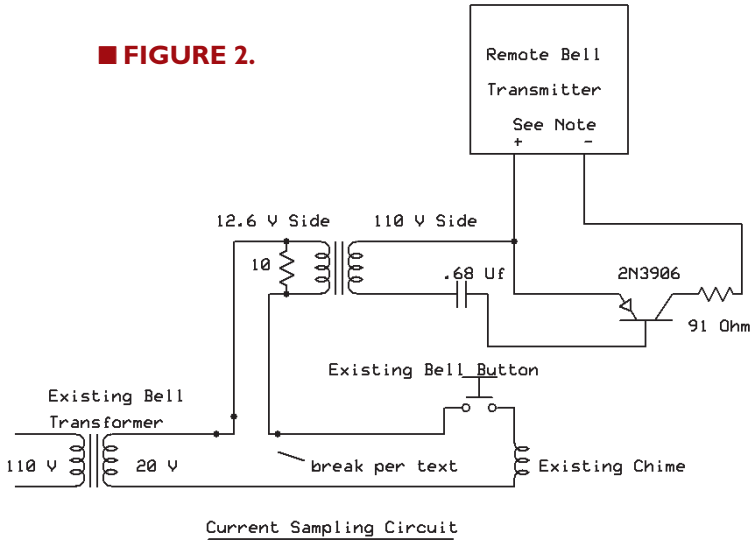
Mailbox Mail Detector: I wired a reed switch across the pushbutton on the remote unit. I attached a magnet to my mailbox door which swings past the switch upon the door opening. Now, I am notified electronically each time the postman makes a delivery.

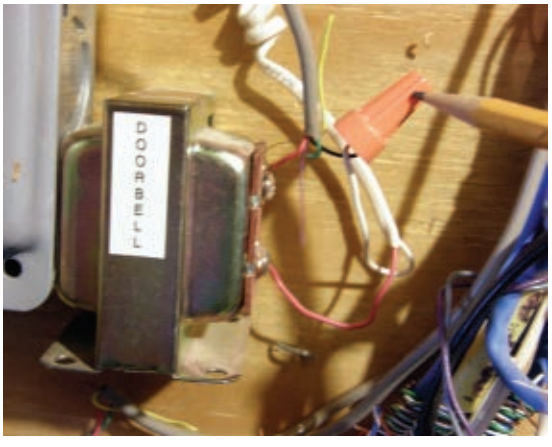
■ FIGURE 1.



■ FIGURE 3.

■ FIGURE 2.

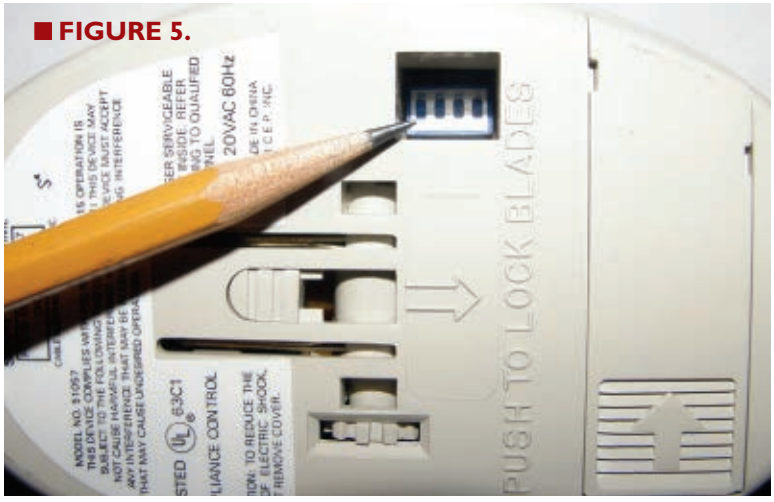




■ FIGURE 4.

Since the doorbell/remote units have code settings, I can designate each one for a certain function. To differentiate one alarm from another, I had to set each of the transmitter/receiver units to its corresponding code. Refer to **Figures 5** and **6**.

■ FIGURE 5.



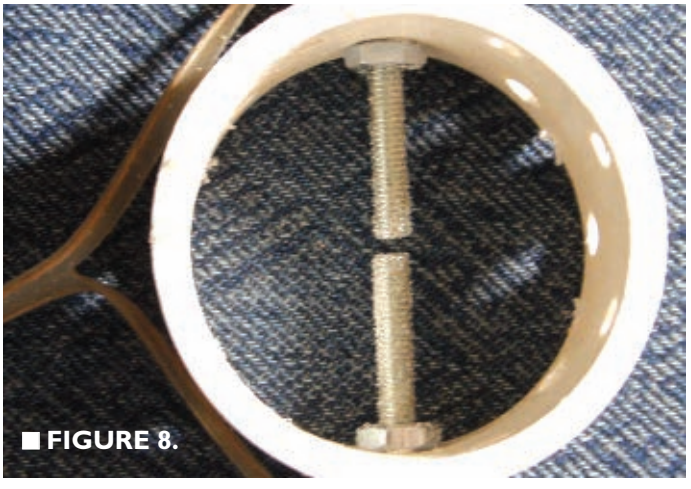
■ FIGURE 6.

The following discussion and procedures provide detailed steps for creating some of the projects discussed here.

Fabrication of the Flood Alarm

1. Cut approximately 1″ of 2″schedule 40 PVC pipe.
2. Draw a line around the circumference of the pipe (about 1/8″ from the bottom).
3. Using a 3/16″ drill, make a hole through the pipe that intersects the line at both sides.
4. Make holes around the circumference in a Swiss cheese pattern (see **Figure 7**).
5. Cut four V notches in the bottom of pipe.
6. Using 10-32 x 2″ bolts, place a nut approximately 1/2″ on each bolt.
7. Place a spade lug on each of two wires that are three feet long.
8. Crimp or solder the lugs to facilitate a good connection.
9. Thread each bolt through the lugs.
10. Fasten each lug with 10-32 nuts (one on each side of lug; leaving 1″ past the connection).
11. Thread the bolts through the pipe on both sides (into the holes you drilled opposite each other).
12. Leave approximately 1/8″ between the bolts and fasten them to the pipe with nuts (see **Figure 8**).
13. Connect the free end of the wires to the transmitter as detailed in the sidebar.

■ FIGURE 7.



■ FIGURE 8.

Connecting to Any Wireless Unit

My original doorbell is a unit that is no longer available, so I purchased a currently-available one to insure compatibility with the designs going forward. The unit I purchased is a Heath Zenith (shown in **Figure 9**; see Parts List for part #) that I feel is representative of today's devices. This unit comes with two transmitters, so it can be used for dual functions by setting the code switches to play different chimes for each event (i.e., a flood or mailbox alarm).

In order for the unit to function as a water alarm, I had to slightly modify the circuit. The modification was simply placing a resistor across the flood sensor. I found the value for it using a decade box. (See my article "The Decade Box Revisited" in the April 2014 issue of *Nuts & Volts*).

The decade box was placed across the sensor (which is wired across the button switch) and adjusted until the bell chimed (see **Figures 10** and **11**). The value was then increased to stop the chime. A suitable resistor was then substituted for the value found with the box.

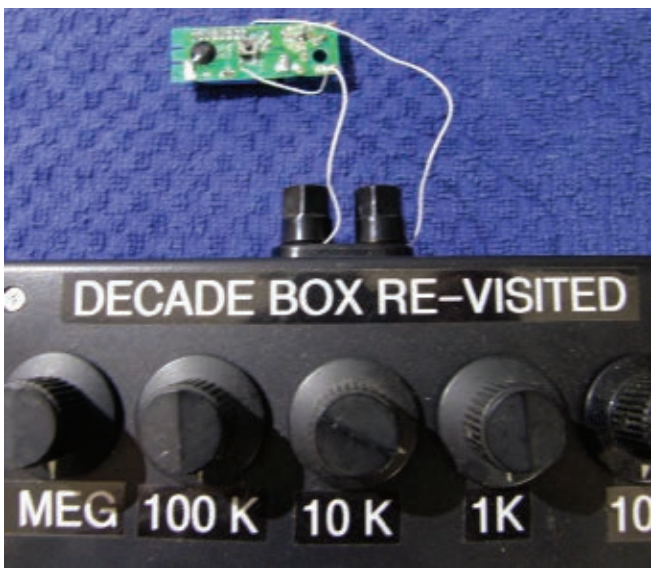
I also discovered the value made the circuit a little unstable, so I increased it to obtain optimum performance. The resistance chosen was 33K ohms. This lowered the circuit impedance to a value that worked when the sensor was placed in water, but made the circuit silent when the water was removed.

■ FIGURE 9.

DESCRIPTION	PART #	SUPPLIER
2N3906	276-1604	RadioShack
91 ohm resistor	022-91	Parts Express
10 ohm resistor	271-132	RadioShack
.68 µF capacitor	027-408	Parts Express
12.6V transformer	273-1352	RadioShack
33K resistor	271-1129	RadioShack
Heath/Zenith bell	16963961678	Home Depot
Misc wire, etc.		RadioShack

Radio Shack.com
Parts Express.com
Home Depot

PARTS LIST



■ FIGURE 10.

■ FIGURE 11.

I also purchased a Harbor Freight (#97004) wireless doorbell to do a comparison (see **Figure 12**). The bell worked (as a water sensor) with a value of 2.7K across the button. Since this value differs significantly from the previous value, I put a potentiometer (or decade box) across each individual circuit to determine the best working value.

The Harbor Freight unit was by far the least expensive one I tested (about $9 with a coupon). Its working range was also excellent (160 feet), but it lacked AC power and relied on two AA batteries. It also had no code switches, so was more prone to outside interference.

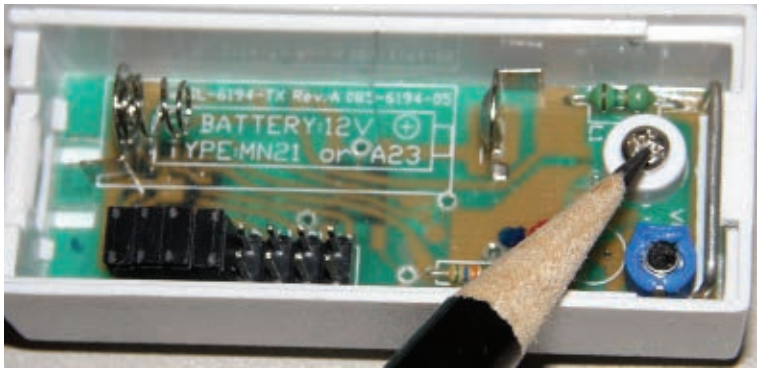
■ FIGURE 12.

How to Open the Heath Transmitter and Modify It

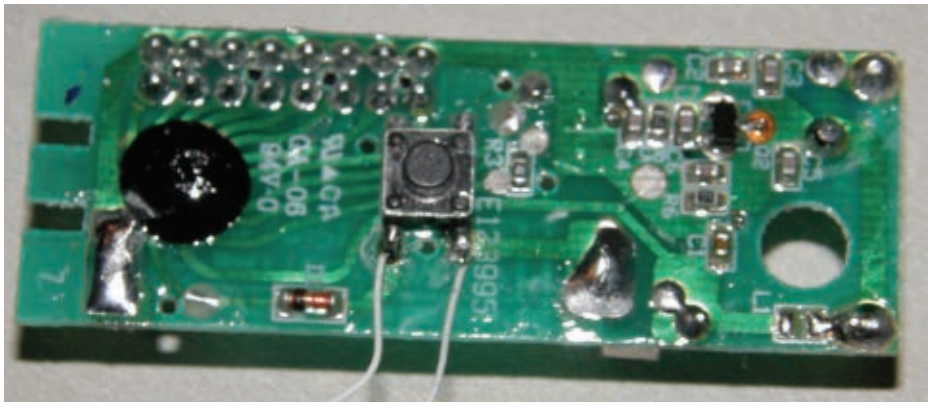
To open the transmitter, follow these steps:

1. Remove the back cover by carefully prying it off.
2. Locate the screw (**Figure 13**) and remove it.
3. Carefully lift out the board and flip it over to reveal the switch.
4. Solder the leads to the contacts (**Figure 14**). I use wire wrap wire to make connections since it's durable and very thin. This makes it easy to route out of the unit and still close the case.

In order to strip the wire, I apply a hot soldering iron and move it along the wire. I also tin the leads (by applying some solder) before soldering them to the circuit.



■ FIGURE 13.



■ FIGURE 14.

Building and Connecting the Control Board for an Existing Doorbell

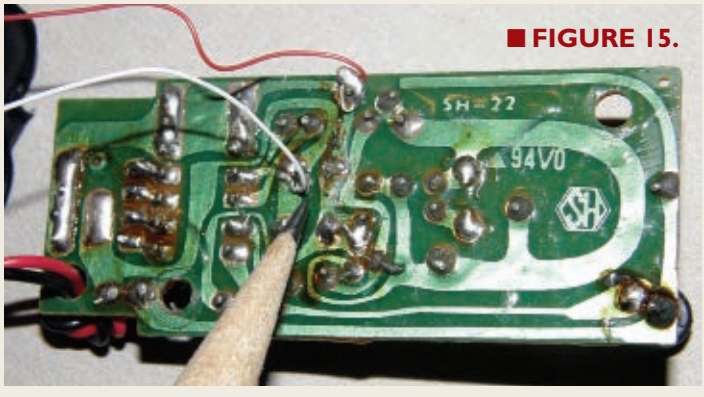
As mentioned previously, I used an external board to interface my existing doorbell circuit (refer back to the schematic) with the new circuits I purchased. You can fabricate this board with any convenient wiring method (e.g., perfboard, printed circuit board, wire wrap, etc.). Wiring is not at all critical. Make the connection to your board by following these steps:

1. Before connecting the board, use a voltmeter to ascertain the polarity of the wires.
2. Connect the (+) positive-most terminal to the emitter of the transistor (the same lead connected to the transformer).
3. Connect the (-) negative lead to the 94 ohm resistor.
4. Attach the board to the transmitter with tie wraps or tape.
5. Make the other connection (to the house bell circuit) by breaking one lead from the house bell transformer's secondary wiring (the end connected to the bell wire).
6. Connect this lead to the 12.6 volt side of the (117V to 12.6V) transformer.
7. Connect the remaining 12.6 volt lead to where the wire was removed (on the house bell transformer).

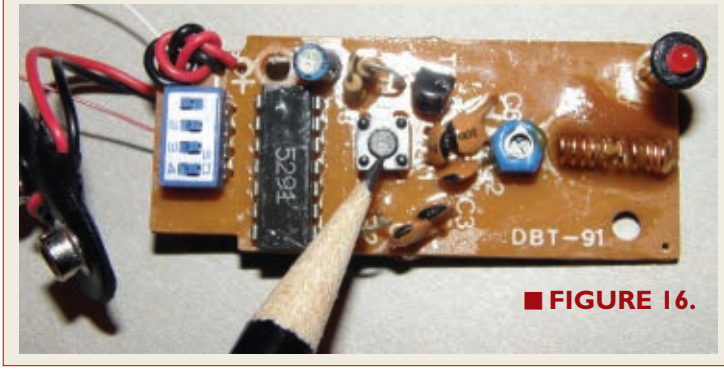
Remote doorbell circuits can be used for a number of projects. Use your imagination! Even if you don't find as great a bargain as I did, you can still find them for less than $25. So, here is my challenge to you: How many uses can you find for these units? NV

TRANSMITTER CONNECTION

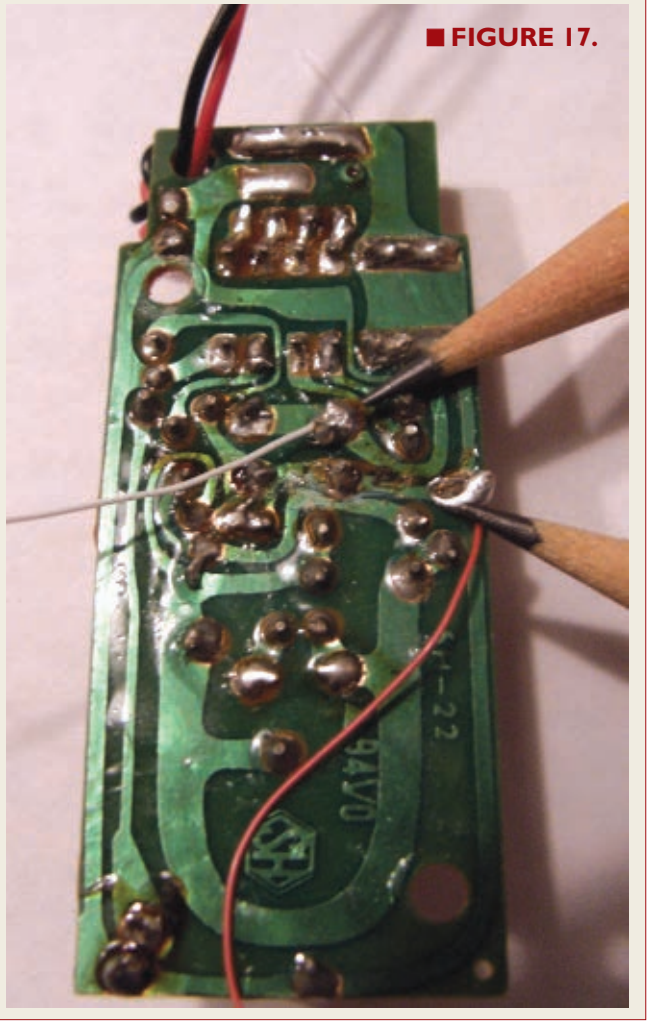
Connection to the transmitter is made by wiring across the button contacts (see **Figures 15**, **16**, and **17**). When soldering the leads, make sure that you don't bridge adjacent foil patterns with solder. If necessary, follow the pattern to a point that is spaced farther from the adjacent foil and connect there.



■ FIGURE 15.

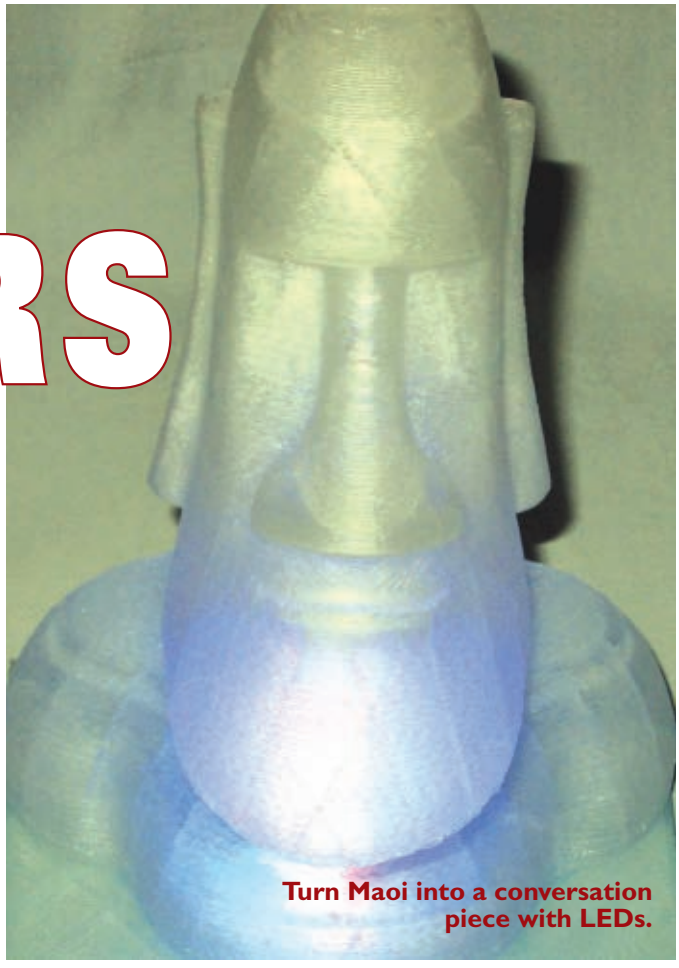


■ FIGURE 16.


■ FIGURE 17.

By William Pippin

USB, LEDs, and SOME SENSORS



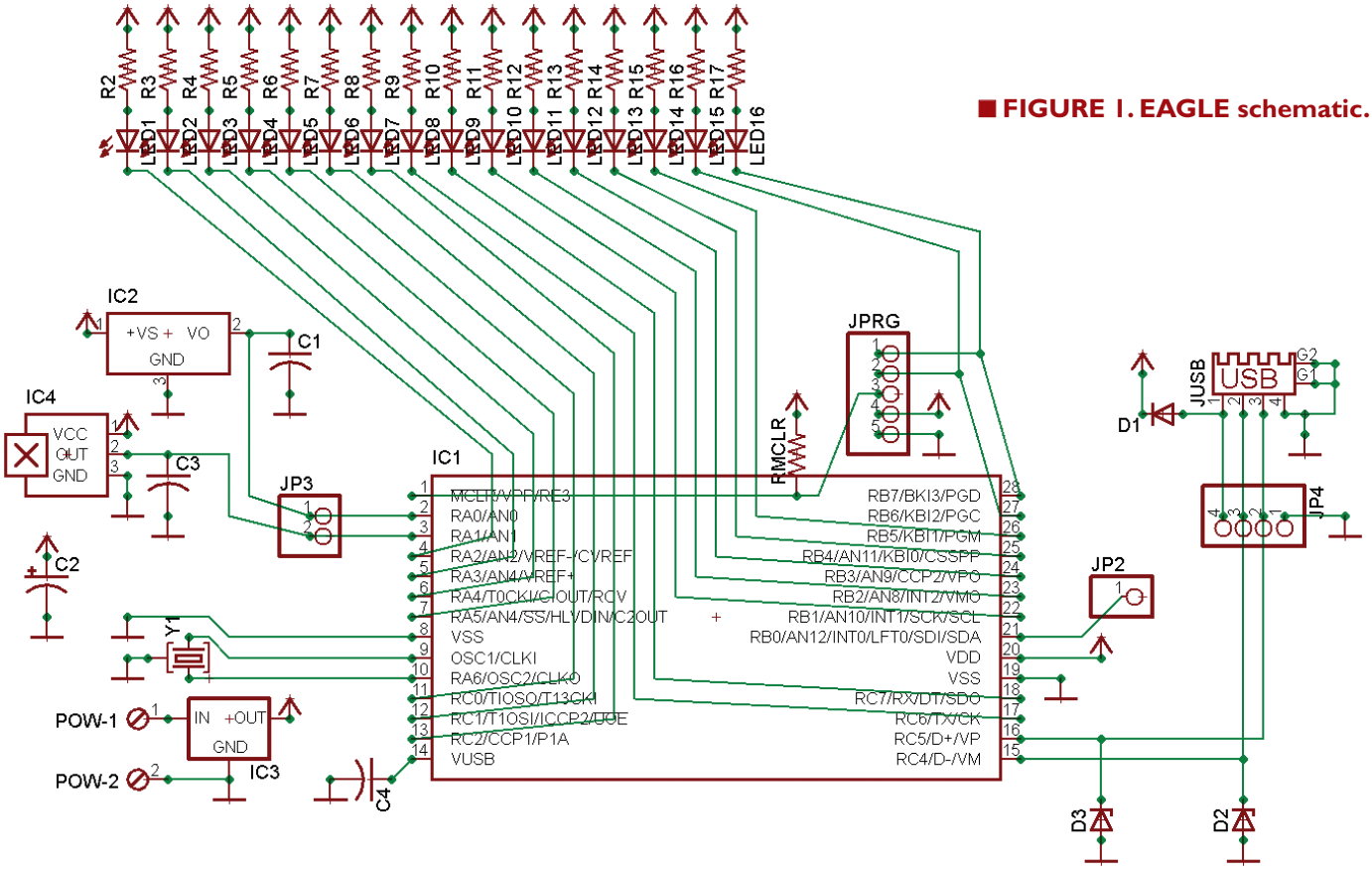
Turn Maoi into a conversation piece with LEDs.

Back in November 2013, I presented a USB keyboard interface in an article. I like to build on things that I know will work so, in that sense, this project is similar to the previous one. The main difference between them is that instead of focusing primarily on an input device, this one centers on an output device.

This project has 16 individually addressable LEDs which can be programmed to your liking: in firmware for stand-alone mode; or controlled via a PC through the USB connection. For added fun, there is a temperature sensor and a Hall sensor. Turning LEDs on and off is fun and all, but why not have a sensor or two that can be useful for something other than the usual light show? With this design, you can use temperature and magnetic fields to wow and amaze your friends.

Let's jump right into this project. Check out the EAGLE schematic view, followed by a pictorial layout design.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/august2014_Pippin.**



■ **FIGURE 1. EAGLE schematic.**

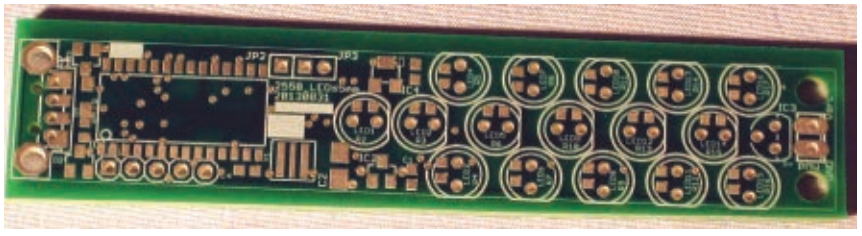
■ **FIGURE 2. Layout.**

ITEM	VALUE	DEVICE	NOTES
C1,C3	0.1 µF	CAP 0603	Noise reduction
C2	4.7 µF	CAPPOL 3216-18W	Current boost for LEDs
C4	0.47 µF	CAP 0603	Required for USB communications
D1	1N4148	DIODE-SOD323-W	USB power protection diode
D2,D3	3.9V	ZENER DIODE SOT23	USB data line protection
IC1	PIC18F2550_28W	PIC18F2550	Main control IC; can substitute PIC18LF2550, PIC18(L)F2450, PIC18F25K50, or PIC18F24K50
IC2	LM50	LM50	Analog output temperature IC
IC3	LP2950ACZ-5.0	LP2950ACZ-5.0	5 VDC regulator
IC4	A1301LH	A1301LH	Analog output Hall sensor
JP2	DNP	PINHEADER 1X01	Test/AUX point
JP3	DNP	PINHEADER 1X02	Test/AUX point
JP4	DNP	PINHEADER 1X04	USB optional through-holes
JPRG		PINHEADER 1X05	*To allow in-circuit programming
JUSB		CON-USB_A_SMD	
LED1-16		Any color 5 mm LED	
POW	DNP	MPT2	Battery Connection
R2-R15	100 ohms to 1K ohm	RES 0603	
R16,R17	10K ohm	RES 0603	*To allow in-circuit programming
RMCLR	39k	RES 0603	
Y1	20 MHz	Ceramic resonator	

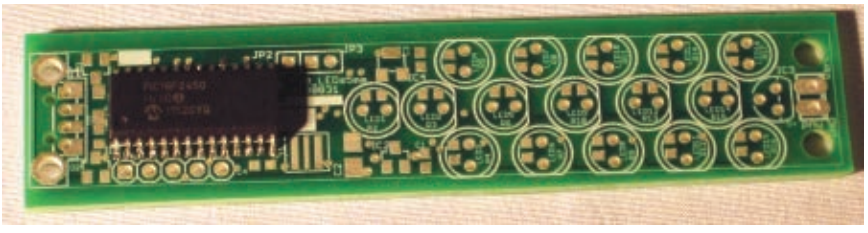
PARTS LIST

■ **FIGURE 3. The original work-in-progress prototype. (Boy, is it messy!)**

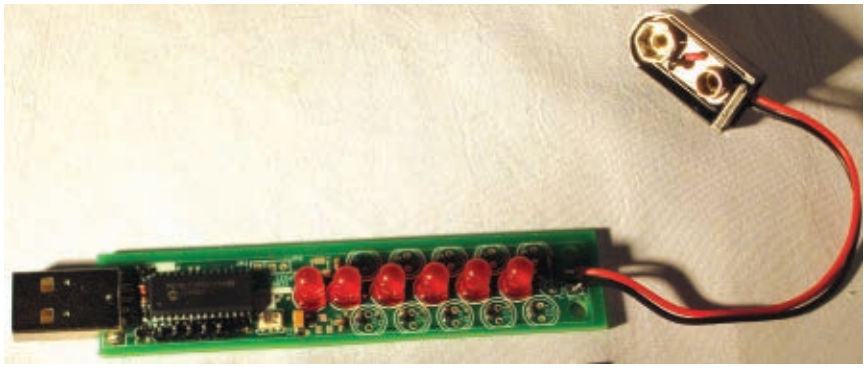
■ **FIGURE 3A. Prototype in plastic enclosure**



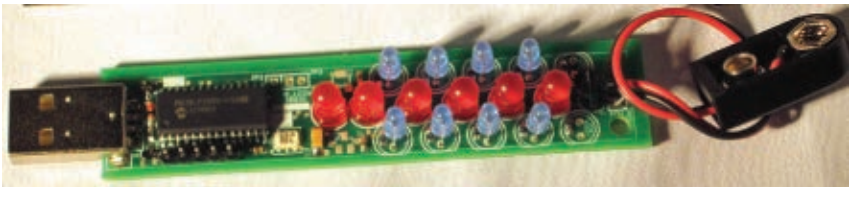
■ **FIGURE 4. The unpopulated production-quality PCB.**



■ **FIGURE 5. First step of PCB assembly; Microchip PIC installed.**



■ **FIGURE 6. PIC support components and center LEDs added.**



■ **FIGURE 7. PCB side LEDs finish the assembly.**

Design Rules

I've created rules for my circuit design that save money and time, and use resources wisely:

- Use parts that I have tested and that I know function properly.
- Whenever possible, use parts that I already have.
- Add extra parts or tasks (e.g., pads, test points, etc.) in a logical way that allows for future variations. I've found that when I spend time to design and test something, I get to know the design very well. So, why create something new when I can build on something that I am already comfortable with?
- Save money whenever possible. Being frugal without sacrificing quality makes sense to me and saves dollars for my clients.

Hardware

Starting at the far left of the layout and moving to the right, I'll explain the rationale behind some of the parts you see in the figures. The USB connector (JUSB) is a surface-mount part that has four vias that overlap the SMT pads. That violates a design rule! I added holes so the modified design could use several of the SMT products available for Type-A connectors, as well as some of the through-hole versions. This allows the design to use a wide range of vendor parts for the USB connector. If you have ever tried to source these, you know how much of a variation in price there can be.

There is nothing apparent in the layout design for the Microchip PIC (IC1) that might seem to allow us to use a wider range of these particular microcontrollers, but there is. The original design was for the PIC18(L)F2550 which has 32K of programming space. The PIC18(L)F2450 part has the same footprint with 16K of program space. There are some savings there if your program doesn't need all that.

Since these USB-enabled parts were released, there are newer drop-in replacement parts available that do not need external clocks (Y1) to function. Namely, these would be the 'K' series parts from Microchip. The 'J' series parts can work, but this design does not provide for an extra capacitor or voltage source that this series would need. I didn't overlook these; I just didn't plan on using them in this design. Besides, the PIC18F25K50 part is plenty cheaper than the original 2550 part.

Remember: The part documentation states that the PIC18LF25K50 does not have a built-in regulator to power the USB connection; however, the PIC18F25K50 does. This circuit design only supports the internal USB regulator parts.

There is no reason to waste space using 5 mm through-hole LEDs when SMT LEDs take up considerably less space. (Refer to the self-imposed design rules.) There are several factors to consider here:

When soldering SMT components, don't worry so much about seeing 100% of the soldering scene. Rely on the fact that a good solder joint flows in a particular way that changes how light is reflected. Learn this art and it will make SMT soldering just a bit easier.

- 5 mm LEDs are cheap and plentiful. You can find them in many surplus shops.
- 3 mm LEDs fit too.
- Since they are through-hole, we can place the LEDs on top or even on pigtails if needed. There are many options.
- For most, the SMT parts are not as easy to change out should they stop working or if you simply want a different color LED in that particular location.
- Through-hole parts have added mechanical strength because of how they are mounted on the PCB (printed circuit board). SMT parts assume and provide very little mechanical resistance.

The final items to consider are the voltage regulator (IC3) and its jumper (POW). The previous USB keyboard design had these items too, but they were rarely needed. That PCB didn't drive any loads by design; it only needed enough current to power the PIC and some switches. The power provided by the USB connection is just fine for that. This design has a bunch of LEDs and those can draw a lot of current depending on the design. Additionally, this PCB is much more useful than the keyboard design as a stand-alone PCB. The 5 VDC regulator allows for an alternate external power source to be connected so that the PCB can function without a PC.

Speaking of LED current, keep in mind that PICs are typically designed to source only about 10 mA and sink about 20 mA of current. That is why all the LEDs use negative logic (a digital 0, 0VDC) to turn on the individual LEDs. This allows the PIC's internal transistors to be used in their most current-efficient configuration. However, this does not mean that the PIC will like driving all the LEDs at full current for very long without getting hot. Of course, this doesn't mean that it cannot. It just means that either we design or program to limit the illumination time of all the LEDs, or we add a heatsink and fan.

I assigned the LED series resistor a value that is slightly higher than it needs to be in order to drop the effective current a bit. If you do the math — assuming a 5 VDC source — typically, you would need about a 100 ohm resistor for each LED. In practice, any resistor value from 100 ohms to 1K ohm will work without much reduction in LED brightness.

I tend to starve LEDs of current in most designs that use them as simple event indicators. I would probably use a 3K ohm resistor which I also used on the previous keyboard circuit. This time, we probably want the LEDs to

Make surface tension your friend when working with SMT. Expect small parts to move and learn to use this to your benefit, not to your frustration.

be bright. You can adjust these values as you see fit. As long as you can see the LED and don't make the PIC sink too much current, everything should be fine.

This design will not populate LED15 and LED16. If they are connected and you hook up an in-circuit programmer to the program jumper (JPRG) and try to program, those LEDs will demand more current than most in-circuit programmers can handle, thereby ruining most programmers.

Make sure you remove the LEDs before you program. Another option: Use a 10K ohm resistor to connect the two LEDs. Doing so usually reduces the current sufficiently to allow the in-circuit programmers to function. If you choose not to remove the LEDs before you program, they will function but will be noticeably dimmer than the others.

Firmware

That should cover all of the physical aspects of this circuit. Let's move on to the firmware. The firmware is quite basic, as is pretty much always the case with good code. The firmware does not follow any formalistic standards. It was written as each part of the functionality was added in order to keep variables and the logical flow easy to understand. Clean code is the best thing that one programmer can give to another programmer.

I do not claim that this is the best code in the world. In fact, it is what I generally call "quick and dirty code" or "Get-R-Done" code (spoken in the voice of Larry, the Cable Guy, of course).

The firmware is divided into three basic sections, listed from the beginning of the file "2550USBLEDs_20121020.c:"

- Global device definitions
- Support functions
- The main function

The first two sections are not useful in the context of this article. The main routine contains all of our upper functionality. It starts out initializing the device by configuring ports, the analog-to-digital converters (ADCs), turning off the LEDs, initializing the USB connection subsystem, and waiting a moment or two. It's not the best code, but it works. You must respond to any new USB connections within the spec time. That is why the *usb_task()* function is called from time to time within all of the code.

That brings us to the main *while* loop. By way of a verbal block diagram, the first 10 or so lines are the repeated housekeeping activities — like allowing the USB functionality time to work, blinking an LED to show that we are alive, and clearing out the LEDs after extended periods of non-activity.

Next, we get our temperature data or our Hall magnetic data based on what mode the device is running in. There are four modes defined so far:

- *DISP_MODE_NONE:* Does nothing unless commanded via the USB port.
- *DISP_MODE_RANDOM:* Randomly sets the LEDs on or off.
- *DISP_MODE_TEMPDATA:* Sets the LED output based on the present temperature value acquired.
- *DISP_MODE_HALLDATA:* Sets the LED output based on the present Hall value acquired.

The last longwinded section of the main function handles the incoming USB commands. This section requires far more code than displaying and acquiring the sensor data does. However, it is no more complex. Its only functions are making sure there is a connection to the USB and there is data waiting on the line, as well as seeing if any of the incoming data matches one of our supported commands.

The commands are pretty simple. The important ones turn the individual LEDs (A-P and a-p commands) on and off, set the modes mentioned previously (W-Z commands), and produce a help screen ('?' command). That's the main function and its infinite *while* loop.

The support functions called by the main loop are pretty straightforward. They validate data, perform some calculations, and either return some data or turn LEDs on or off. Don't get me wrong; I am not over-simplifying things. We are all hobbyists here, so I don't want to over-explain that which is obvious.

As presented, this design requires a driver file to be loaded on a PC. The driver files (provided by CCS, Inc.; **www.ccsinfo.com**) produce the compile with which this project was built. The PIC can be programmed to use any driver that you write your own code to utilize. So, technically, you are not limited at all. However, if you want to use the design as-is, you must use the drivers provided, as well as those that are available for download.

I will make a firmware version that also allows the use of the HID interface. These types of devices generally do not require additional drivers. In my November 2013 article, I described what might need to be done if you use a really old PC.

Troubleshooting

As for troubleshooting, most — if any — issues will concern getting the PC drivers to install correctly. Once that is done, if the device responds to commands issued via a communications port connection (via the virtual USB-to-serial drivers), that is a good sign the device is

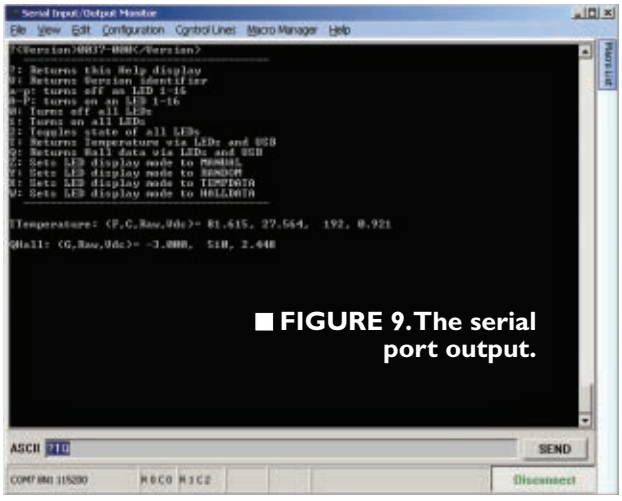
FIGURE 8. Initial LED test in random mode.

powered and mostly working. If the LEDs do not illuminate or sensors do not return data, uninstall and reinstall them.

On the off chance that a component is bad, replace it. If you cannot get the device to connect via USB and the drivers have been installed, make sure that diode D1 is installed correctly. If it is backwards, the device will not work via the USB power connection alone. If you have diode D1 installed backwards and have an external power supply connected, it would still work, and the USB connection would be active. However, in this scenario, you would be supplying five volts DC to the PC's USB data link. In most cases, nothing would happen, but many laptops only provide 3 VDC on their USB devices. In this case, you would probably damage the USB port. Needless to say, I recommend installing diode DI correctly.

I hope this design is simple and straightforward enough to be helpful. I further hope that the logic and rationale used were more beneficial than the core functionality of the design. The physical design is quite simple, but the firmware can make this little circuit do a lot of useful things. Once you add a PC to the USB connection, who knows what a little imagination can bring!

Remember, there is a lot more that you can do with this simple circuit. As long as you stay within general safety and device limitations, you should be just fine. NV



■ **FIGURE 9. The serial port output.**

I designed a simple enclosure for this project. It's nothing fancy, and you will have to trim it to fit because I made the design quite tight in order to ensure a snug fit. I uploaded the project files for those who have access to a 3D printer. The files are at **www.thingiverse.com/thing:157512**. Also, the examples of using this PCB to illuminate translucent 3D prints are courtesy of the **thingiverse.com** shares, as well.

 Continued from page 27

intuitive analysis. Q-Scape is especially helpful in three-phase system analysis as each phase can be displayed on a dedicated tab. For those desiring a larger display area and even higher display resolution, the HDO8000 supports extended desktop operation with a WQXGA (3840x2160 pixels) DisplayPort 1.2 video output. Q-Scape tabbed displays may then be viewed on a larger unit while another program (e.g., MATLAB) is viewed on the smaller oscilloscope display. While several models are in a higher price range, there is a 16 digital channel MSO option that is priced at US$2,800.

For more information, contact:
Teledyne LeCroy
Web: **www.teledynelecroy.com**

EXPANSION OF LOW COST PIC32MX1/2 SERIES

Microchip Technology, Inc., has announced a new family of PIC32MX1/2 microcontrollers (MCUs) in 256/64 KB Flash/RAM configurations. These new MCUs are coupled with comprehensive software and tools from Microchip for designs in digital audio with Bluetooth®, USB audio, graphics, touch sensing, and general-purpose embedded control. The MCUs are an expansion to the popular PIC32MX1/2 series of low cost small footprint 32-bit microcontrollers. They now offer larger Flash and RAM options with a feature-rich peripheral set at a lower cost. The PIC32MX1/2 boasts a wide variety of features including I2S for digital audio, large memory configuration, and 83 DMIPS performance for executing Bluetooth audio and advanced control applications; CTMU for capacitive touch sensing; eight-bit PMP for graphics or external memory; a 10-bit 1 Msps 13 channel ADC; and serial communications peripherals with the PIC32MX2 series supporting USB-device/host/OTG functionality.

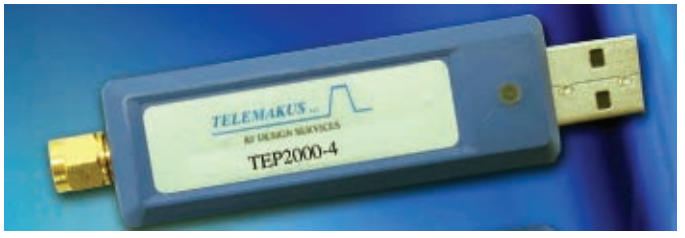
PIC32MX1/2 MCUs are targeted for low cost applications in the consumer markets such as Bluetooth speakers, consumer music-player docks, noise-cancelling headsets, infotainment systems, clock radios, and entertainment system sound bars, as well as touch screens with buttons and sliders, and USB device/host/OTG applications. PIC32MX1/2 MCUs are available in 28-pin QFN, SOIC, SPDIP, and SSOP packages, and 44-pin QFN, TQFP, and VTLA packages. The PIC32MX1/MX2 series is supported by Microchip's free MPLAB® X IDE and the MPLAB XC32 Compiler for PIC32. Application-specific development tools that support the PIC32MX1/MX2 series include the PIC32MX270F256D plug-in module for

the Explorer 16 development board (US$35) and the PIC32MX270F256D plug-in module for the Bluetooth audio development kit (US$25).

For more information, contact:
Microchip
Web: **www.microchip.com**

DIGITAL PHASE SHIFTER USB RF MODULES

Saelig Company, Inc., announces the availability of the Telemakus TEP2000-4 and TEP4000-5 digital phase shifters — the smallest USB microwave phase shifters available. These laboratory quality phase shifters have a minimum phase range of 360 degrees (400 degrees typical) with 12-bit, 0.25 degree resolution. The TEP2000-4 operates over the bandwidth of 1 GHz to 2 GHz and the TEP4000-5 operates from 2 GHz to 4 GHz. Maximum RF input for linear operation is +6 dBm but the devices can handle up to +20 dBm. Typical insertion loss is only 4 dB to 6 dB. The phase shifters contain 0.5 GB of Flash memory used for installation files, test data, drivers, documentation, and user-defined information. Telemakus USB RF modules can be easily controlled with any Windows XP, 7, or 8 PC via USB with the simple included graphical user interface (GUI). A software API is also provided so the modules can be built into a LabVIEW-based test setup.



Applications for Telemakus modules include phased array antenna testing and other RF equipment testing. Weighing less than an ounce each, Telemakus test devices represent the latest technology in low cost/portable test equipment, and can be easily transported with a laptop for RF equipment field servicing. Each unit's GUI is resident in onboard Flash memory making them "plug and play" for Windows-based PCs.

Multiple Telemakus USB RF modules can be connected and controlled to build up a complex system of RF functions that include switches, attenuators, amplifiers, signal generators, vector modulators, phase shifters, frequency doublers, and power detectors at a fraction of the cost of benchtop equipment. Prices start at US$385.

For more information, contact:
Saelig Company, Inc.
Web: **www.saelig.com**

I/O

A New Add-on Adapter for the Raspberry Pi

By Ben and Tom Kibalo

For quite a while now, we have sought the ultimate add-on interface board to enhance some Raspberry Pi (RasPi for short) projects we want to pursue, including a mobile webcam and a wireless sensor network. For such projects to work, it's essential that our RasPi add-on meet several particular criteria.

For starters, its microcontroller must be easily programmable. This is important because we need to be able to offload the RasPi from the more mundane control/monitoring functions, leaving it free to do vision processing and/or web control.

The add-on would also need a healthy mix of I/O — both digital and analog interfaces — and an expansion port capable of addressing custom electronics. The adapter shouldn't be larger than the RasPi to make packaging easy and it would also lend itself to better mechanical stability when mounted to the RasPi (to handle vibrations).

Another necessary feature is easy access to the RasPi general-purpose input/output (GPIO), so as not to block onboard cable connectors (that is, the camera). Finally, in the interest of green design, the add-on should cycle and control power to the RasPi remotely using a real time clock calendar (RTCC) to facilitate shut-down and/or power-up.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/august2014_Kibalo**.

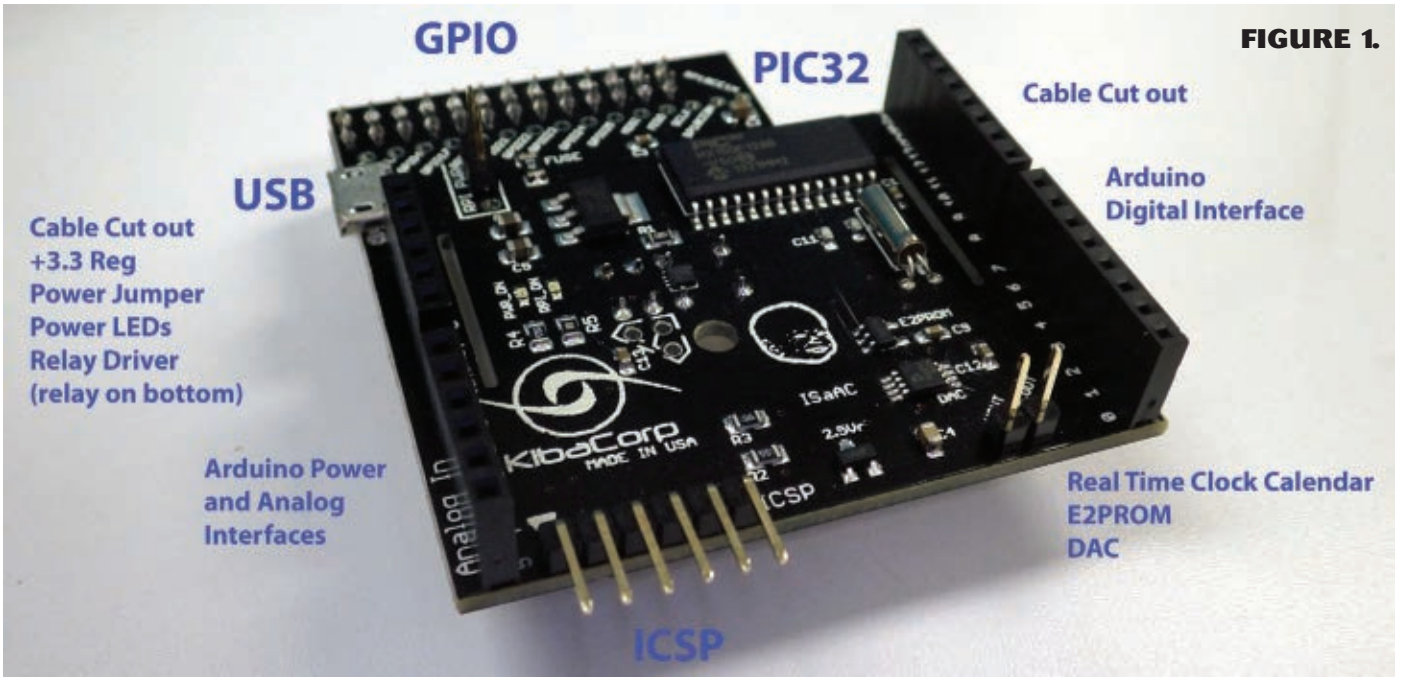


FIGURE 1.

Several electronic add-on adapters that are commercially available might have provided our solution. We had considered the top candidates to be the Gertboard and the aLaMode (both processor-based add-ons), but we had to rule out the Gertboard because of its size. Then, we focused our attention on the aLaMode. It offered both a programmable microcontroller and an Arduino-style expansion interface. Its size is right for the RasPi, and it has fully accessible GPIO. However, because cables must be routed around the board, the RasPi card cable connections are blocked. Another issue with aLaMode is that its Arduino connectors (A0 to A5) can easily touch the bare metal of the RasPi RJ45 which is a potentially dangerous situation in a mobile environment. Finally, aLaMode has no remote control power.

We decided to build our own add-on adapter which we designated the "ISaAC" (**I**deal **S**ystem **a**nd **A**pplication **C**ircuit) to address all of our criteria directly (see **Figure 1** showing ISaAC and **Figure 2** showing ISaAC mounted onto the RasPi). Our final ISaAC design resulted in an onboard, relay-based power control to the RasPi, a Microchip 32-bit processor, a DAC for signal generation, EEPROM (for nonvolatile storage access as needed), an Arduino-compliant electronic interface, open slots onboard to pass through a RasPi ribbon connector, and an RTCC for precision timed-based control in a compact SMT package.

While designing the ISaAC, another thought occurred to us besides hardware. The ISaAC uses a Flash programmable Microchip PIC32. How about developing the ISaAC to contain a pre-programmed environment — one in which all onboard functionally is available through

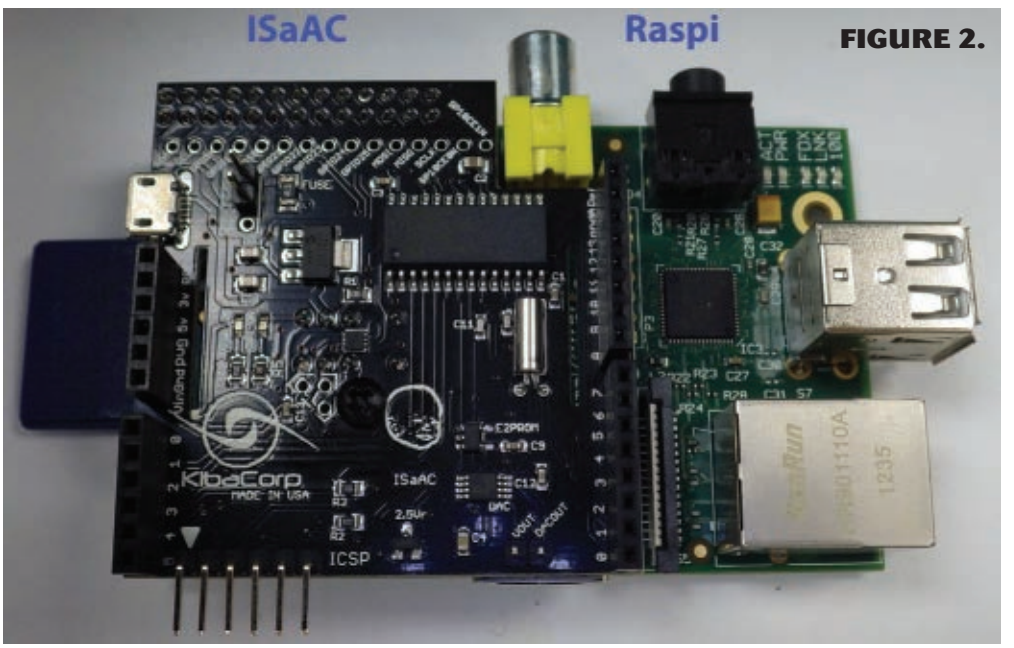


FIGURE 2.

an API (Application Program Interface)? The API would support any custom hardware debugging without the need for developing special test code. It would also facilitate rapid development of any project application software.

We composed the API using an ASCII command/response serial interface, thereby allowing easy direct viewing of any results. The benefit of the serial port is that it allows for available application software (Teraterm, HyperTerminal, and Minicom) that can work with the API directly. Once an API is learned in this way, it can be easily invoked within Python using an ISaAC API library. The ISaAC API (currently 19 commands) is discussed in depth in the **Reference Section** sidebar, as well as a description of ISaAC Arduino format pins.

The ISaAC board comes fully assembled, pre-programmed, and tested. It supplies power to the RasPi using a micro USB. Its +5V power-only interface helps minimize the number of USB cables required for the RasPi. Power to the RasPi is through a +5V relay controlled by ISaAC located on the bottom of the ISaAC board.

The default setting for the relay is Power On. Selecting relay power for the RasPi uses an onboard jumper setting. Turning off the relay power requires an API command U — see the ISaAC *Technical Reference Manual* — which (when used with the RTCC) allows an orderly shutdown and restart of the RasPi. Once powered down, the ISaAC/RasPi current consumption drops considerably.

The ISaAC has a Microchip In-Circuit Serial Program (ICSP) interface that allows reprogramming of the ISaAC Flash to accommodate API revisions.

In this article and in others that will follow, we will guide you through an increasingly challenging series of DIY projects using ISaAC. To assist with these experiments, we have made a technical reference manual (mentioned earlier) available at the article link. This manual describes the electronics and API operations in detail, the ISaAC Python API library, and the ISaAC API MPLAB X library project (for those who want to familiarize themselves with ISaAC API implementation).

Let's Get Started

As mentioned, the ISaAC uses the RasPi serial port on the GPIO for API communications. First, we must configure this administrative port so the system software can access it. Once we've accomplished this task, we will work with the low level API to test our hardware. RasPi Minicom is a good way to become familiar with the API

Reference Section

1.0 — ISaAC Pinout Reference

Reference	Digital I/O Pin	API Designation	Function
D0	0	00	Programmable digital I/O
D1	1	01	Programmable digital I/O
D2	2	02	Programmable digital I/O
D8	8	08	Programmable digital I/O
D9	9	09	Programmable digital I/O or PWM
D10	10	10	Programmable digital I/O or PWM
D11	11	11	Programmable digital I/O
D12	12	12	Programmable digital I/O
D13	13	13	Programmable digital I/O

Reference	Digital I/O Pin	API Designation	Function
A0	0	14	Analog In 10-bit ADC/Programmable Digital I/O
A1	1	15	Analog In 10-bit ADC/Programmable Digital I/O
A2	2	16	Analog In 10-bit ADC/Programmable Digital I/O
A3	3	17	Analog In 10-bit ADC/Programmable Digital I/O
A4	4		I²C Bus Data
A5	5		I²C Bus Clock

API Commands	Function
?	Query state of digital input pin
A	Set pin to analog input/read
CR	Read RTCC clock
CS	Set RTCC clock
D	Output to DAC
ER	Single EEPROM read
EW	Single EEPROM write
H	Set output pin high
I	Configure pin for digital intput
L	Set output pin low
M	Help menu
O	Configure pin for digital output
P	Set pin to PWM output
Q	Build script
T	Set alarm
U	Pi power cycle on alarm
V	View EEPROM
X	Execute script
Z	Z Precision pulse

2.0 — ISaAC API Reference

ISaAC communicates to the Pi over the GPIO serial port at 192008N1. API commands are ASCII upper case characters and are echoed back for easy debugging. Command verification from ISaAC is simply 'A' (acknowledge) or 'N' (not acknowledge). There are 19 unique API commands.

while testing and debugging hardware before any coding.

Once we are satisfied with proper hardware operation, we will use the Python lganguage and the ISaAC Python Extended API library to develop the application example. We will step you through a process for building a RasPi servo control system. In this example, the servo position is set by changing a potentiometer setting. A blinking LED indicates the program is active.

Installing ISaAC

1. Prepare the RasPi for the ISaAC board. We must reconfigure its serial port (ttyAMA0) because — by default — it is set as the administrative terminal. Open the */boot/cmdline.txt* file for editing.
2. Remove the following:

```
console=ttyAMA0, 115200 kgdboc=ttyAMA0,
115200
```

3. Save and close the file.
4. Preventing the RasPi from outputting boot information over the serial line is an important step. Otherwise, the serial buffer can get overwhelmed by all the data. This can make it difficult to successfully send commands to the ISaAC initially. Open */etc/inittab* for editing.
5. Place a hashtag (#) in front of the following line that reads:

```
TO:23:respawn:/sbin/getty -L ttyAMA0115200
vt100
```

6. Use the following command to initiate the RasPi's shutdown:

```
sudo shutdown now -h
```

7. After shutdown, attach the ISaAC board to the RasPi. ISaAC supplies power to the Pi through the GPIO, so only one micro USB is required (the one that connects to ISaAC). Place the jumper on the RPI PWR spot on the ISaAC and attach the micro USB power cable to it.
8. The ISaAC uses a red LED for +3.3V power indication and a blue LED for relay on/off indication. Both boards should power-up now, and these LEDs should be on. Make sure that your RasPi is functioning normally via your method of choice (i.e., SSH or HDMI port).
9. Open Minicom using the following command:

```
sudo minicom -s
```

You must request superuser permissions to make changes to the configuration and access the onboard serial port. If you don't have Minicom installed, type the following command:

```
sudo apt-get install minicom
```

10. Now that Minicom is open in configuration mode, choose the serial port setup menu item and then press A to change Serial Device to */dev/ttyAMA0*.
11. Press E to adjust the serial port settings. Scroll through the menu with A and B until you find a baud rate of 19200. Ensure the *Current:* line at the top reads 19200 8N1.
12. Press enter to leave this screen, and press enter again to go to the main menu.
13. Click Save setup as dfl. To make this the default setting, click Exit which will place you into the active Minicom terminal.

Running ISaAC

Here are a few precautions before we proceed:

• If you are breadboarding your circuits with a separate power supply, make sure to link the power supply ground and ISaAC ground pins. For most simple experiments, the ISaAC should be able to supply power to the breadboard.

• For routine purposes, only use the ISaAC board to supply power to the RasPi. If independent cables are used to power the RasPi, remove the RPI PWR jumper and make sure both boards use the same power source. Failure to do so risks permanent damage to both boards.

Testing the Hardware With API

Test 1 — LED

Let's use a classic "HELLO, WORLD" example to test the API under Minicom.

1. Plug in an LED using the appropriate resistor for any of the peripheral pins on the ISaAC (D0-D13 or A0-A3). Refer to the circuit diagram in **Figure 2**.
2. Open the Minicom terminal as a superuser.
3. Let's assume we have connected to pin 1 on our

FIGURE 3.

ISaAC board. Set the pin to an output by typing O01 into the terminal; you should receive an "A" as a response.

4. Type H01. The light should now turn on (if it hasn't already). Type L01 and the light will turn off. We set pin 1 to an output with the first command; we set that output to high (on) and then we set it to low (off).

These pins can assume three states: high output, low output, or input (otherwise known as tri-state GPIO). All of the pins on the ISaAC can be commanded in this way. A few of the pins have additional special functions:

- Set pin to output – O, then two digits for pin
- Set pin to high – H, then two digits for pin
- Set pin to low – L, then two digits for pin

Test 2 — ANALOG ADC

Measuring digital states is very useful, but having the ability to measure analog signals adds another dimension of utility. The ISaAC has four pins that have 10-bit analog-to-digital conversion (ADC). These pins are labeled A0 to A3. An example circuit can be created with a thumb potentiometer following the **Figure 3** schematic.

Pins A0 to A3 are designated as pins 14, 15, 16, and 17 in the API. We will use pin A0 for this example.

1. Set the pin to an input using the command *I14*.
2. Set the pin to an analog state using the command *A14*. This sets the pin and returns the current value as a 10-bit integer.
3. Play around with the potentiometer and send the *A14* command again to get a feel for the different readings. You should see a range from 0 to 1023. This function is limited to the pins listed above.

    - Set analog pin to input – I, two digits for pin (14 to 17)
    - Set pin to analog mode, read state – A, two digits for pin (14 to 17)

Test 3 — SERVOS

Standard servos usually run at 50 Hz. Sending a pulse duration between 1 ms and 2 ms at 50 Hz will cause the servo to rotate fully from one direction to the other. ISaAC uses a pair of pins (9 and 10) that can provide servo control signals using the API.

A servo has a three-wire connection: white for control signal; red for +5V; and black for ground. Use the 5V power and GND from the ISaAC to power the servo. You will also need an op-amp chip to drive the servo (I used the MAX908CPA quad op-amp). The LM124 works just as well (same pinout). See the op-amp in **Figure 5**.

No special level shifters are required in this case because the comparator is supplying power to the servo.

Note: Some pins on the ISaAC are not 5V tolerant.

1. To move the servo using 50 Hz frequencies with pin 9 connected to the servo white wire, use the following API command:

```
Z09000500000001500
```

This is pin 09 with a frequency of 00050 Hz, a delay of 0000 ms, and a pulse of 001500 ms.

2. To move the servo through its range of motion, change the Z parameter's setting of the last four digits to 1200 and 1800.

    - Set pin to output – O, either pin 09 or 10
    - Send precision pulse command:

```
"Z" (two digits pin) (five digits
frequency Hz) (four digits delay
in ms) (6 digits pulse width in µs)
```

Installing Python Libraries

1. We set up the GPIO serial port for use earlier in this article. Now, we must install the Python serial library to use it. To do so, execute the following command as a superuser:

```
sudo apt-get install python-serial
```

2. ISaAC comes with an extended Python API Library (*issaccomm2.py*) for running ADC, normal commands, and servo position under Python. To use this within your Python programs, you must install it within your Python Code folder. The library provides Python 2.x support for the ISaAC board in RasPi applications. Automatic error handling and built-in help are included in the library.
3. To use this library, type:

```
import isaaccomv2
```

4. To invoke help at any time, type:

```
help (isaaccomv2)
```

5. Several API functions are provided in this version of the library, including:

    - Simple API command execution
    - ADC measurement (by pin)
    - Analog out
    - Pin assignment (digital or analog)

- PWM (by pin)
- EEPROM data storage and EEPROM data recall
- RTCC time set and read
- Alarm set and configure wake event (including power-up of RasPi)
- An API script build/recall/rewrite capability

To perform API calls in Python, use the form *API_com (string)*. This function sends API commands directly to the ISaAC board. Here are some examples:

- Make ISaAC's pin 1 a digital output: *"API_com("O01")*
- Set it high to turn on LED: *API_com("H01")*
- Set it low to turn off LED: *"API_com("L01")*

The command *API_ADC(pinna)* accepts a two-digit string for the pin number to measure (14, 15, 16, 17) and returns a two-component array (for example, echo, *ADC_data = API_ADC("14")*. The first component is the echo response from the ISaAC board. The second component is a 10-bit integer value for the analog value of that pin.

For servo control, the API provides a higher level function servo position (*servo value, servo pin)*. This function positions the servo to the *servo_value*, using pin 9 or 10 to provide servo control signaling. An example using pin 9 and the ADC data value is:

```
echo, ADC_data = API_ADC ("14")
servo position (9, ADC_data)
```

There's also an API function that blinks a light on a pin for a designated time using the form *blink (pin, time_val)*.

You already used the core API with Minicom to test and debug your hardware. Now, you can use this extended API within a Python file or directly with IDLE GUI. API usage is straightforward, supports project hardware tests, and makes code development easier by letting the user focus on the software debugging only.

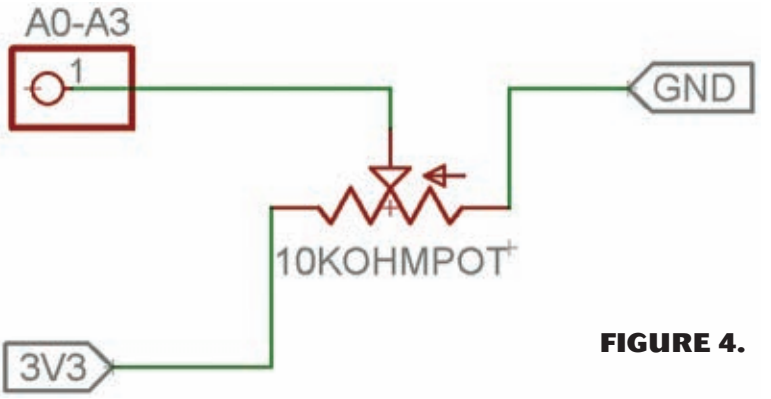


FIGURE 4.

Servo Control Using Python

Let's run this project using Python to control servo position. The position of the servo will be set by reading a potentiometer.

1. Turn the pot CW, and the servo position will change in the CW direction.
2. Turn the pot CCW, and the servo will change accordingly. We will also throw in a blinking LED to indicate that the project is actively running. See the schematic in **Figure 5** again for the entire setup. Go ahead and assemble this circuit.
3. Use the Extended API library for running ADC, normal commands, and servo position as discussed earlier.

A diagram of the prototype is also shown in **Figure 5**.

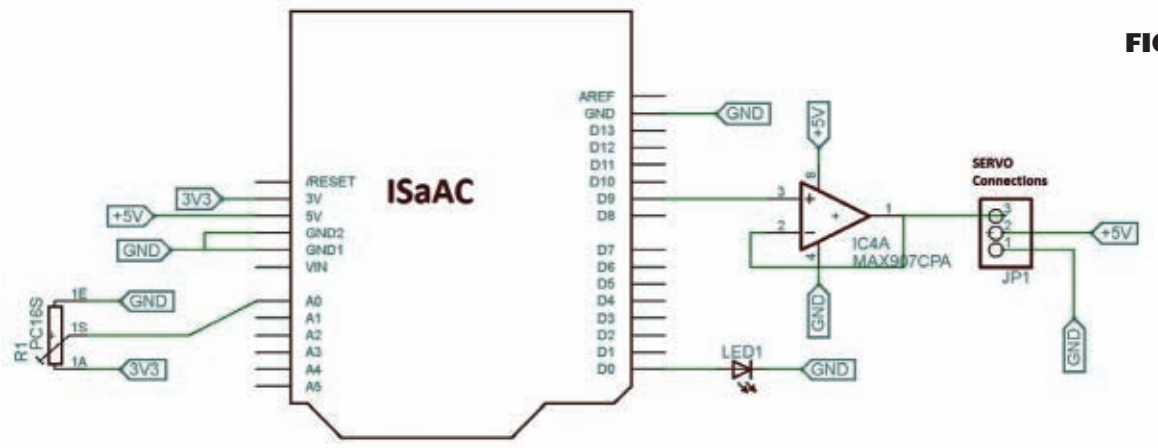
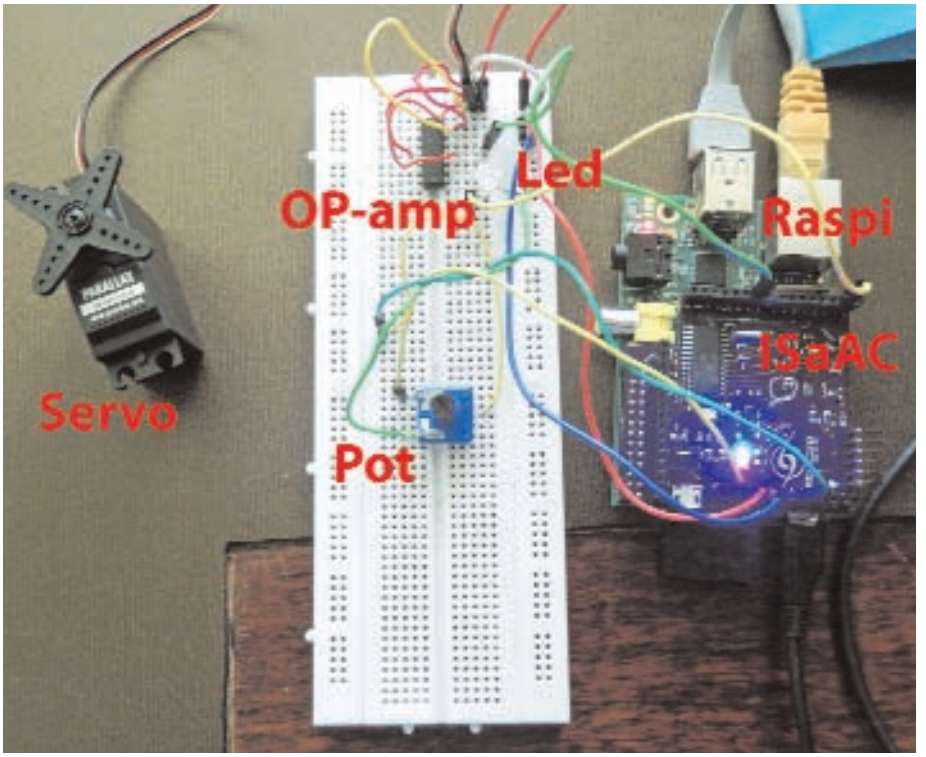


FIGURE 5.

FIGURE 6. Servo control prototype.



ISaAC supplies power to the prototype electronics, servo, and RasPi with one micro USB. A solderless breadboard is used for auxiliary electronics and is connected to the ISaAC/RasPi via breakout connectors on ISaAC.

What's Next?

We covered a lot of detail on ISaAC's capability using the API. There's more to come! This API ensures rapid development. Be sure to download the reference manual for examples on each API command, Flash programming the ISaAC, and more. If you'd like to purchase IsaAC, you can do so from the *Nuts & Volts* Webstore.

We hope to cover a number of exciting applications with ISaAC in the near future, including a video CAM, wireless sensor data logging, and mobile operation. So, stay tuned! **NV**

■ BY JOE PARDUE

The Arduino Classroom
Arduino 101 — Chapter 8: Displaying Information

Last month, we looked at motion and position control using the variable resistance of potentiometers. This culminated with labs that led up to using the turning of a potentiometer knob to control the motion and position of a servomotor.

We've learned a lot of fundamental principles for both computing and electronics, and we have reinforced that learning with hands-on laboratory exercises. This month, we will go very light on new fundamental information and have some fun with a bunch of labs that deal with my favorite thing: blinking LEDs.

We will look at displaying information using LEDs configured as a bar graph and as a seven-segment single character display.

What is Information?

Information is something that informs — it provides data — it settles uncertainty — and it is hard to define without eating its own tail. There is something unknown, then information happens, and then that something is known. Up to now, we have sent information to the Arduino and received information from it using the USB connection to a PC and the Arduino IDE's (Integrated Development Environment) serial monitor. Now, we will look at two simple ways to convey information from an Arduino to someone without needing the USB hooked up to a PC.

First, we'll look at an LED bar graph. You've probably seen these on audio equipment where the number of LEDs lit informs you of something like the audio signal volume. One might also be used as a sort of gas gauge where the number of LEDs lit represents the amount of fuel available.

Next, we'll look at a seven-segment LED like the four shown in **Figure 1**. These can be used to display numbers and — if you make allowances — can crudely display the entire alphabet.

Displaying Magnitude With an LED Bar Graph

Lab 1: Lighting the Bar Graph Segments — Digital

Parts required:

1 Arduino
1 Arduino proto shield and jumper wires
7 LEDs
7 1,000 Ω resistors

Estimated time for this lab: 30 minutes

Check off when complete:

❑ Build the circuit shown in **Figures 2**, **3**, **4**, and **5**. In this circuit, each LED has an individual 1K Ω, but each of these resistors is carried to ground by a bundling of wires that may be somewhat difficult to visualize.

FIGURE 1: *Seven-segment LED time display.*

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/august2014_Smiley**.

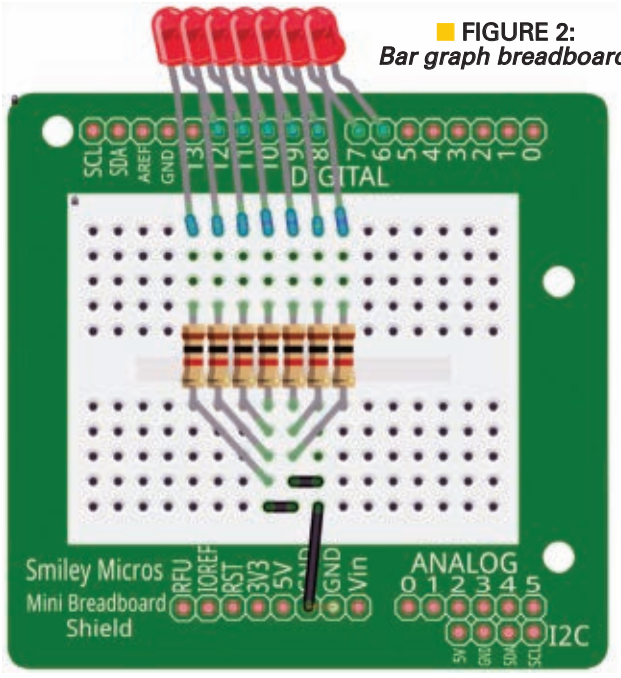


FIGURE 2: *Bar graph breadboard.*

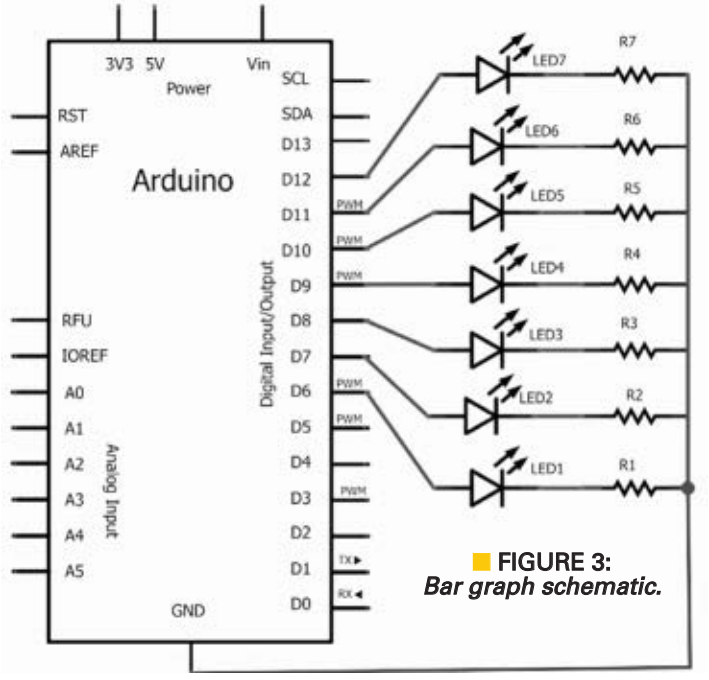


FIGURE 3: *Bar graph schematic.*

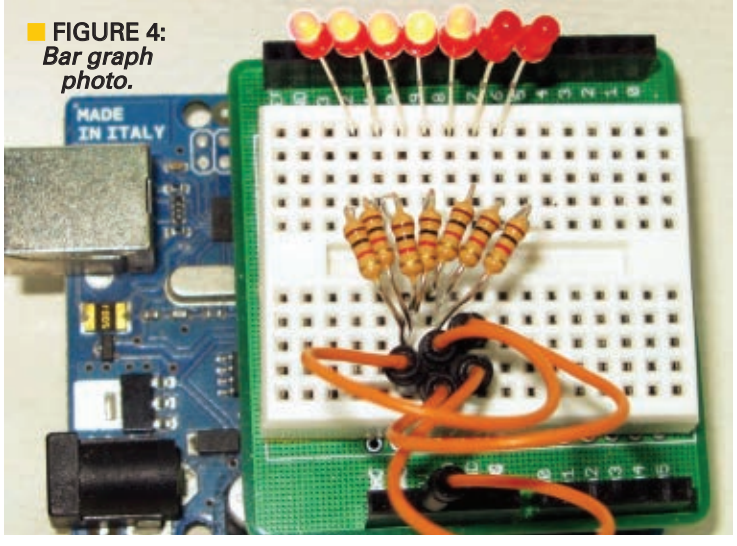


FIGURE 4: *Bar graph photo.*

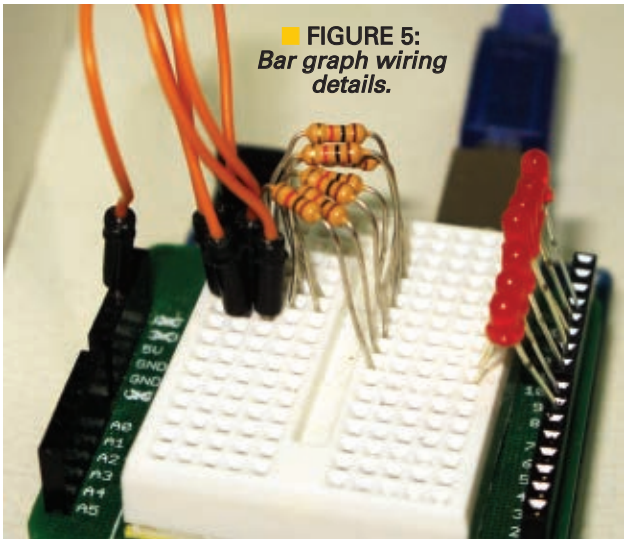


FIGURE 5: *Bar graph wiring details.*

❑ Insert the following program into the Arduino IDE. All code listings are available at the article link.

```
// A101_ch8_bar_graph Joe Pardue 5/31/14

// lowest indication starts with LED 6
// hightest indication ends with LED 12
int low = 6;
int high = 12;

// milliseconds between LED turn ons
#define SPEED 50
#define PAUSE 500

void setup(){
  // Set LED pin modes to output
  for(int i = low; i <=high; i++){
    pinMode(i, OUTPUT);
  }
}

void loop(){
 // scroll LEDs on
 for(int i = low; i <= high ; i++)
 {
   digitalWrite(i, HIGH);
   delay(SPEED);
 }

 delay (PAUSE); // pause between scrolling
                // LEDs

 // scroll LEDs off
 for(int i = high; i >= low; i--){
   digitalWrite(i, LOW);
   delay(SPEED);
 }
```

```
    delay(PAUSE); // pause between scrolling
                  // LEDs
}
```

- ❏ Observe the LED scrolling pattern.
- ❏ Read the following source code discussion to fully understand how this program works and to reinforce the programming concepts shown in this program.
- ❏ Go to the forum on **www.arduinoclassroom.com** if you have any questions.

Source Code Discussion

Let's take a few moments to review each line of this code to reinforce what we have learned so far. If you completely understand all the code, you can skip this discussion. However, if you are the least bit iffy, then read it. The first line in the program:

```
// A101_ch8_bar_graph_data_display Joe Pardue
5/31/14
```

This identifies the name of the source code, the author, and the date the code was released. The next section defines two variables available for use by functions within this module:

```
int low = 6;
    // lowest indication starts with LED 6
int high = 12;
    // hightest indication ends with LED 12
```

We define variables by first indicating the variable type; in this case, they are of the *int* type that — for the Arduino — can contain numbers from 0 to 65535. The first variable is named *low* and set to equal the rightmost LED in our design which is connected to Arduino pin 6. The second variable is named *high* and set equal to the leftmost LED in the design which is connected to the Arduino pin 12.

Each of the seven LEDs is connected sequentially from pin 6 to pin 12. Make certain that you can relate this source code for the pin usage to the illustrations shown in **Figures 2**, **3**, **4**, and **5**. These LEDs can be lit to represent a relative magnitude from 0 (none lit) to 7 (all lit). Since the pin numbering and the LED numbering are offset by 6 (that is to say, the lowest LED is located on pin 6), then a magnitude of 1 would be represented by lighting the LED on pin 6; a magnitude of 2 would be represented by lighting the LEDs on pins 6 and 7; and so on to a magnitude of 7 which is represented by all the LEDs from pin 6 to pin 12.

The next section of code defines two constant values for milliseconds we will use to provide delays between turning on each LED (*SPEED*), and between scrolling them all on and all off (*PAUSE*):

```
// milliseconds between LED turn ons
#define SPEED 50
#define PAUSE 500
```

The first function in an Arduino program is *setup()* where we do things once that need to be done before we run the program. In this case, we set each of the pins connected to LEDs to be outputs so that we can provide current to turn the LEDs on or off:

```
void setup(){
  // Set LED pin modes to output
  for(int i = low; i <=high; i++){
    pinMode(i, OUTPUT);
  }
}
```

Inside the *setup()* function, we have a *for-loop* that is used to set the LED pins to output:

```
  // Set LED pin modes to output
  for(int i = low; i <= high; i++){
    pinMode(i, OUTPUT);
  }
```

The *for-loop* has three components. First is the variable *i* defined as an *int* and set to the value of *low* — the rightmost LED pin. The next states the *i* shall be less than or equal to *high* — the leftmost LED pin position. The last element increments *i* so that on each pass through the *for-loop*, *i* increases in value by one. In this case, the *i* starts out with a value of 6 (low). Then, after each pass through the loop, it increases by one while it is less than or equal to 12 (high). When the *i*++ sets *i* to 13 — which is not less than or equal to *high* (12) — the code exits the *for-loop*.

Within the *for-loop*, we see:

```
    pinMode(i, OUTPUT);
```

We learned that the *pinMode()* function takes an Arduino pin number as the first parameter and then sets that pin to be either an input or an output, depending on the value of the second parameter. We set it to *OUTPUT* which is a constant defined in the depths of the Arduino library that tells the *pinMode* function to set the indicated pin to output.

We use a *for-loop* to run this function seven times. We could have avoided the *for-loop* and written:

```
    pinMode(low, OUTPUT);
    pinMode(low+1, OUTPUT);
    pinMode(low+2, OUTPUT);
    pinMode(low+3, OUTPUT);
    pinMode(low+4, OUTPUT);
    pinMode(low+5, OUTPUT);
    pinMode(low+6, OUTPUT);
```

Either form will work, and which one to use is up to the person who writes the code. As a rule of thumb, you might just write out the operations as shown if you have four or fewer iterations, and use a *for* (or *while*) loop for more than that.

Next, we have the *loop()* function where the Arduino

performs repetitive operations. In this code, we will run a *for-loop* to turn on all the LEDs, pause, then run a *for-loop* to turn off all the LEDs, pause, and repeat forever:

```
void loop(){
 // scroll LEDs on
 for(int i = low; i <= high ; i++)
 {
   digitalWrite(i, HIGH);
   delay(SPEED);
 }

 delay (PAUSE); // pause between scrolling LEDs

 // scroll LEDs off
 for(int i = high; i >= low; i--){
   digitalWrite(i, LOW);
   delay(SPEED);
 }

 delay(PAUSE); // pause between scrolling LEDs
}
```

The first *for-loop* starts with the low (rightmost) LED and turns it on. Then, it sequentially turns on each higher value LED until it gets to the *high* (leftmost) LED. The *for-loop* uses the *digitalWrite(i,HIGH)* function where the first parameter (*i*) is the Arduino pin number to set. The second parameter is *HIGH*, a constant that tells the Arduino to provide a high voltage (+5 volts) to the indicated pin. The next function is a delay set to the constant *SPEED*. The only purpose of that line is to slow down the process so as to animate the LEDs turning on.

After the loop finishes, another delay is called using *PAUSE* to provide another animation effect. The next *for-loop* starts where the first one left off, with all the LEDs lit. It turns them off in the opposite order from how they were turned on. It starts with the high LED, turns it off, decrements the value of *i*, and repeats until it has counted down from high to low, turning off each LED in sequence.

The *digitalWrite()* function is given the pin number and then told to set it *LOW*, which turns off the +5V. After this, there is another *pause* delay, then the loop begins again at the top, and loops forever scrolling the LEDs.

Lab 2: Data Display on a Bar Graph

In Lab 1, we discussed in detail how the source code worked to provide a review of what we have learned so far. Lab 2 takes that program and adds some features so that we can set a given magnitude on the graph — thus providing the user with visual feedback of magnitudes from zero through seven. In this section, we will only discuss those aspects of the new program that differ from the first program.

Parts required:
1 Arduino
1 Arduino proto shield and jumper wires
7 LEDs
7 1,000 Ω resistors

Estimated time for this lab: 15 minutes

Check off when complete:
❏ We will reuse the circuit built in Lab 1 with no modifications.
❏ Enter in the following program into the Arduino IDE.

```
// A101_ch8_bar_graph_data_display Joe Pardue
// 5/31/14

// lowest indication starts with LED 6
// hightest indication ends with LED 12
#define LOWEST 6
#define HIGHEST 12

// milliseconds between LED turn ons
#define SPEED 50

void setup(){

  Serial.begin(57600);
  Serial.println("A101_ch8_bar_graph rev
  1.0");

  // Set LED pin modes to output
  for(int i = low; i <=high; i++){
    pinMode(i, OUTPUT);
  }
}

void loop(){
  // receive a value to set the bar graph
  if(Serial.available()) {
     barGraph(Serial.parseInt());
  }
}

// sets the bar graph LEDs to the indicated
// magnitude
void barGraph(int magnitude){

  Serial.print("magnitude = ");
  Serial.println(magnitude);

  if(magnitude < 8)   // show magnitude if 0
                      // to 7
  {
    magnitude = LOWEST + magnitude - 1;

    alloff(); // turn them all off

    // turn on only to the magnitude
    // indicated
    for(int i = LOWEST; i <= magnitude ;
    i++)
    {
      digitalWrite(i, HIGH);
      delay(SPEED);
    }
```

```
    }
    else // scroll once
    {
      alloff(); //turn them all off
      // turn on sequentially
      for(int i = LOWEST; i <= LOWEST+7 ; i++)
      {
        digitalWrite(i, HIGH);
        delay(SPEED);
      }
    }
  }

  // turn off all the LEDs by setting the pins
  // LOW
  void alloff(){g
    for(int i = LOWEST; i <= HIGHEST; i++){
      digitalWrite(i, LOW);
    }
  }
```

Source Code Discussion

Let's take a few moments to review each line of this code that differs from the program in Lab 1. We will review the serial communication functions used in this program, then look at the *bargraph()* and *alloff()* functions. As before, if the code is clear, then skip this discussion.

Serial Review

This program sets up the serial communications and sends out a notification of what program is running in the *setup()* function as follows:

```
Serial.begin(57600);
Serial.println("A101_ch8_bar_graph rev 1.0");
```

The first line uses the *Serial.begin()* function to tell the serial class that the serial port will communicate at 57600 baud. The second line uses the *Serial.println()* function to send out the program name and revision information on the serial port.

In the *loop()* function, the serial port is scanned for input and that input is used to acquire the data that will be shown on the bar graph by calling the *barGraph()* function. The loop runs until the *Serial.available()* function becomes true, indicating that data has been received on the serial port. It then converts that text data into an integer using the *Serial.parseInt()* function as a parameter of the *barGraph()* function:

```
if(Serial.available()) {
   barGraph(Serial.parseInt());
}
```

Note that using a function as a parameter to a function works the same as using *Serial.parseInt()* as a parameter for *barGraph()*. An alternate way to do this would be:

```
int myInt;

myInt = Serial.parseInt();

if(Serial.available()) {
   barGraph(myInt));
}
```

So, we can do it the longer clearer way by defining an integer variable, then setting that integer to the value returned by the *Serial.parseInt()* function, and then using the variable as the parameter for the *barGraph()* function. Or, we can do the shorter version and use the *Serial.parseInt()* function directly as the parameter to the *barGraph()* function, where the integer returned by *Serial.parseInt()* is provided directly to the *barGraph()* function. This latter way of doing things is very common among programmers.

The final serial function is in the barGraph() function where we use two lines:

```
Serial.print("magnitude = ");
Serial.println(magnitude);
```

The first line sends the string "*magnitude* = ". The second line then converts the magnitude parameter [sent by the *Serial.parseInt()*] into a string that is sent out the serial port, followed by a line return.

The BarGraph() Function

After the *barGraph()* function sends the magnitude out the serial port, it first checks to see if the magnitude is in an acceptable range:

```
if(magnitude < 8)// show magnitude if 0 to 7
```

This checks to see if the magnitude variable is less than eight, meaning it is somewhere from zero to seven. If the magnitude is in the acceptable range, then the first block of code is run and the magnitude is shown by lighting the indicated number of LEDs. If the magnitude is out of range, then the LEDs are scrolled (turned on sequentially, then turned off). If you see the LEDs scrolling, you know you sent an incorrect number.

To turn the LEDs on indicating the magnitude requested, first the magnitude value is converted to a value that indicates the pin number for the first (lowest value) LED on the *barGraph*:

```
magnitude = low + magnitude - 1;
```

If, for instance, the magnitude is three, then we see from the variable definitions at the beginning of the program that low is six, so 6 + 3 - 1 = 8 — which becomes the new magnitude. This will allow the LEDs to be turned on from the low LED on pin 6 to the highest LED on pin 8. Next, all the LEDs are turned off:

```
alloff(); // turn them all off
```

Then, the LEDs are turned on one at a time in the

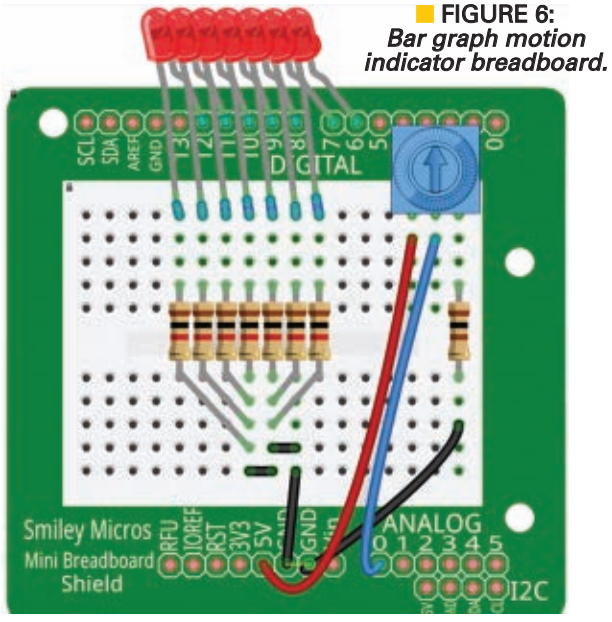
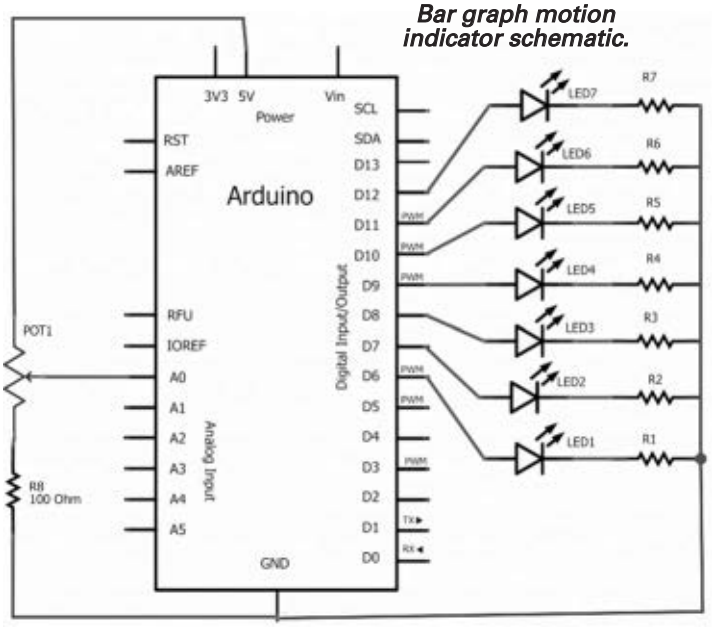


FIGURE 6: *Bar graph motion indicator breadboard.*

FIGURE 7: *Bar graph motion indicator schematic.*



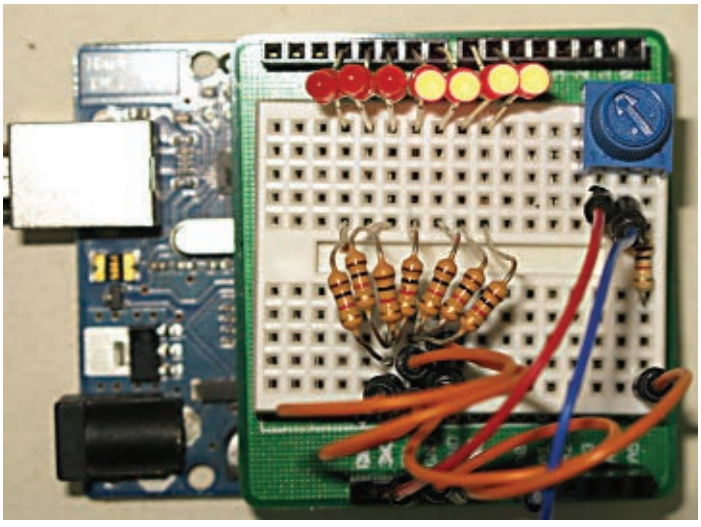


FIGURE 8: *Bar graph motion indicator photo.*

for-loop from the low LED to the high LED in a sequence with a speed represented by *SPEED*. The pin represented by *i* is set to *HIGH*, providing current for the LED to turn on:

```
// turn on only to the magnitude indicated
for(int i = low; i <= magnitude ; i++)
{
  digitalWrite(i, HIGH);
  delay(SPEED);
}
```

That was a lot of reviewing and reinforcing! If you are unsure of any aspect of these programs, don't hesitate to ask questions on the forum. Be sure and mention the chapter number and the lab exercise.

Lab 3: Dial Position Display on a Bar Graph

Parts required:

1 Arduino
1 Arduino proto shield and jumper wires
7 LEDs
7 1,000 Ω resistors
1 Potentiometer
1 100 Ω resistor

Estimated time for this lab: 30 minutes

Check off when complete:

- ❏ Reuse the circuit built in Labs 1 and 2, to which we will add a potentiometer with a current-limiting resistor as shown in **Figures 6**, **7**, and **8**.
- ❏ Since we have spent so much space reviewing the software, let's assume competence and just refer the reader to the source file *A101_ch8_bar_graph_motion_indicator.ino* at the article link.
- ❏ Run the code and move the potentiometer dial to verify that the code is working.

Displaying Characters With a Seven-Segment LED Display

In the bar graph labs, we used seven LEDs to display a relative magnitude. We could use similar techniques to turn the seven LEDs into a display to show alphanumeric data (see **Figure 9**). These seven-segment displays are primarily used to display numbers, but if you are willing to accept some crude approximations (and memorize a few weird

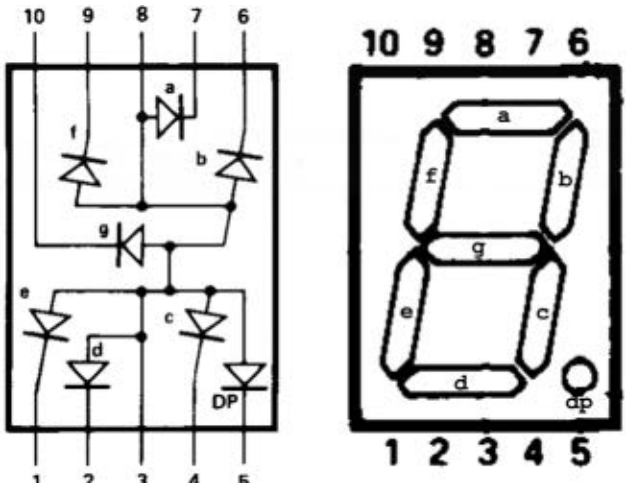


FIGURE 9: Seven-*segment LED schematic.*

substitutions — see **Figure 10**), you can display the full alphabet. Indeed if you are truly generous, you can turn this device into the world's smallest moving message sign — something we will do as our final lab in this chapter. First, though, let's get familiar with the device.

Note that although this is a seven-segment LED, it actually has eight segments when you include the decimal point that is implemented for using several seven-segment LEDs side by side to show a decimal place. **Figure 9** shows a schematic for the seven-segment display we will be using. There are actually many variations on this schematic with different pinouts.

Two basic types are available: common cathode and common anode. We are using the common anode type as shown in **Figure 9**. Each LED has a letter designation, and each of these is connected to a specific pin (for example, letter a is connected to pin 7). In our case, we also have pins 3 and 8 connected to the anode (positive voltage). To use this device, we will connect pin 3 to a 100 Ω resistor, and each of the LED cathode (negative) pins to Arduino pins that we will configure as outputs. When we want to turn an LED on, we will output LOW (ground) so that current flows from the +5, through the 100 Ω resistor, through the LED, and into the Arduino pin to ground. When we want to turn the LED off, we will set the Arduino pin HIGH (+5V) so no current will flow.

The design we will use for this lab has one major weakness in that we will channel all the current through a single resistor so that the more LEDs are lit, the weaker the current will be and the dimmer the LED will appear. You won't be able to use this in direct sunlight, but for learning purposes it will serve nicely.

We will use a design for a character set that will give us readable images of all numeric and most alphabetic characters as shown in **Figure 10**.

Generally, these are used only to display numerals, which you can see are clearly readable. The alphabetic

FIGURE 10: Seven-*segment font.*

characters require a bit of forgiveness on the part of the viewer, but can be used to covey text information for those willing to accept the obvious limits.

Lab 4: The Seven-Segment LED Display

Parts required:

1 Arduino
1 Arduino proto shield and jumper wires
1 Seven-segment LED
1 100 Ω resistor

Estimated time for this lab: 30 minutes

Check off when complete:

❏ Build the circuit shown in **Figures 11**, **12**, and **13**. Notice this circuit uses a single 100 Ω resistor between the seven-segment LED anode pin 3 and the five volt power supply.
❏ Enter the following program into the Arduino IDE:

```
// A101_ch8_7seg Joe Pardue 6/10/14
// lowest indication starts with LED 6
```

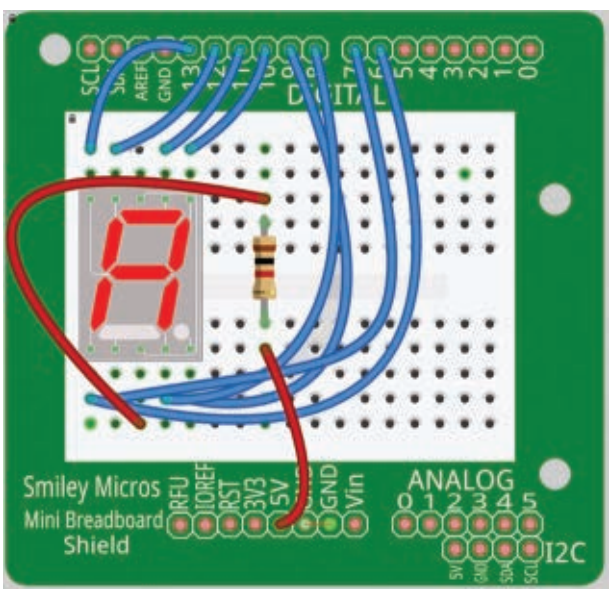


FIGURE 11: Seven-*segment breadboard.*

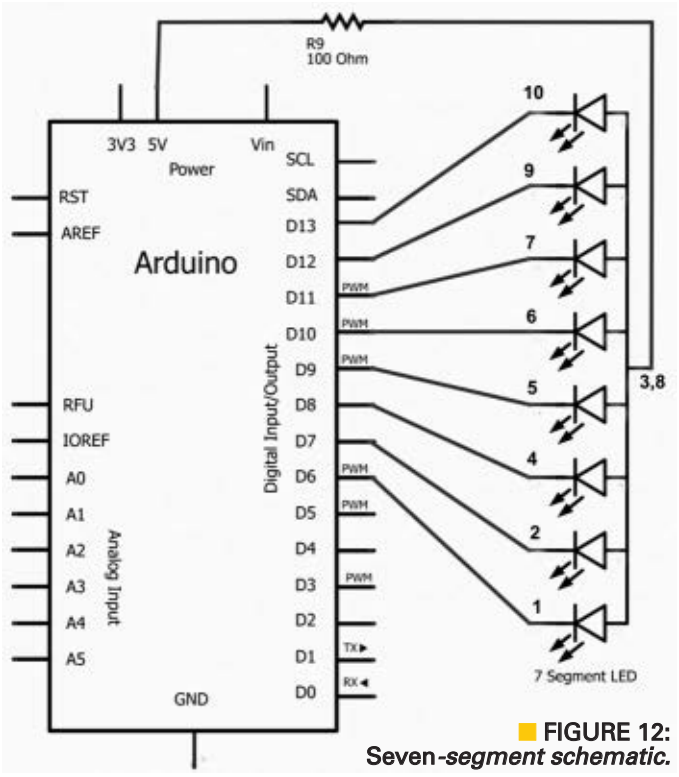


FIGURE 12:
Seven-*segment schematic.*

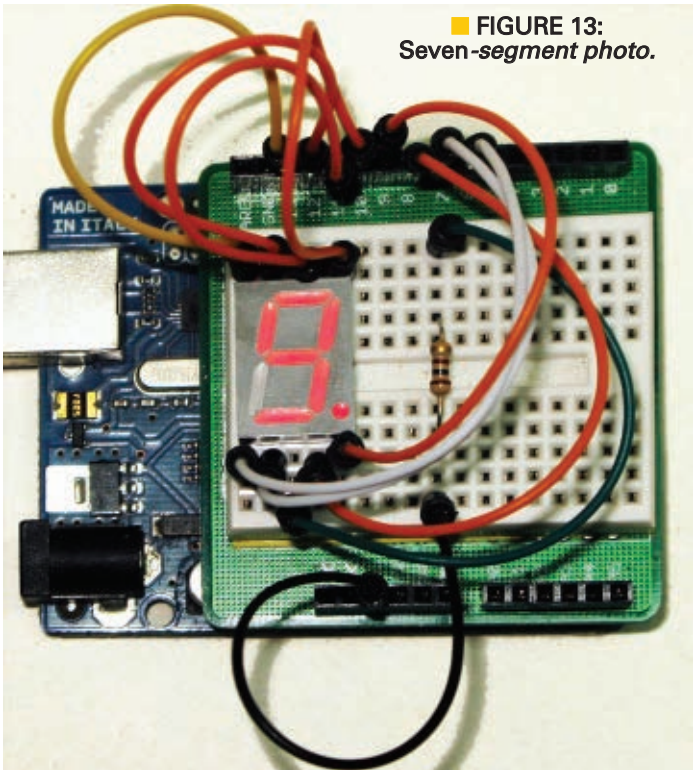


FIGURE 13:
Seven-*segment photo.*

```
// hightest indication ends with LED 12
int low = 6;
int high = 13;

void setup(){
  // Set LED pin modes to output
  for(int i = low; i <=high; i++){
    pinMode(i, OUTPUT);
  }
}

void loop(){
 // scroll LEDs on
 for(int i = low; i <= high ; i++)
 {
   digitalWrite(i, LOW);
   delay(250);
 }

  // scroll LEDs off
 for(int i = low; i <= high ; i++)
 {
   digitalWrite(i, HIGH);
   delay(250);
 }
```

❏ Run the program, and you will see each of the seven segments turn on in sequence and then turn off in a reverse sequence.

Lab 5: The World's Smallest Moving Message Sign?

I've built a library (*sevenSegLED*) that we will use to create our moving message sign. It's available with the other downloads. Note that we have not gone over some of the programming concepts contained in the library code and while you have no need to look at the code to use it, if you do decide to check it out, be forewarned that it is well advanced of our studies to-date. You can, of course, look at the source code for the library. Just don't be too put off by some of the potentially arcane things therein. It will all make sense one day as you continue your Arduino studies.

Several years back, I published a similar application for another device using the same title. That title wasn't quite true then and it still isn't. Displaying a single character at a time and having some of them being just plain not like the version you grew up with ... well, it is a stretch. However, if you are willing to learn those off-beat characters and accept that some will be displayed upper case while others will be displayed lower case, then you will have one really cheap way to get text out of an Arduino.

Parts required:
1 Arduino
1 Arduino proto shield and jumper wires
1 seven-segment LED
1 100 Ω resistor

Estimated time for this lab: 30 minutes

Check off when complete:
❏ We will reuse the last circuit unmodified for this lab.
❏ Unzip the A101_ch8_supplemental.zip file and note the

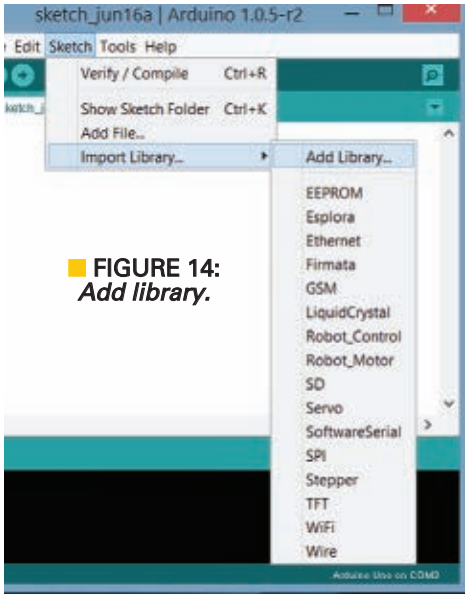


FIGURE 14: *Add library.*

FIGURE 15: *Select sevenSegLED.*

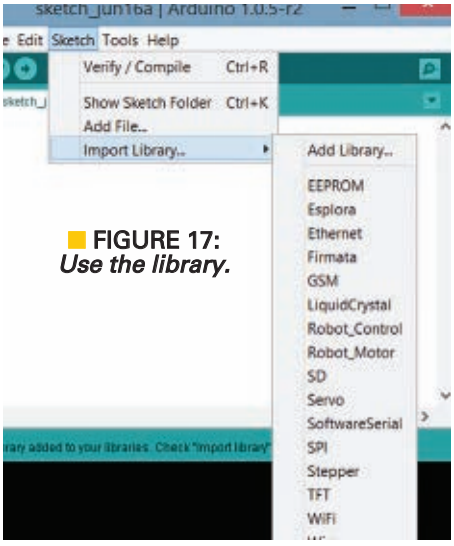


FIGURE 17: *Use the library.*

FIGURE 16: *Library added.*

FIGURE 18: *Adds include file.*

location of the sevenSegLED.zip file.

- ❏ In the Arduino IDE Sketch/Import Library tab, click the 'Add Library' button as shown in **Figure 14**.
- ❏ This will open a file browser window like **Figure 15**. Browse to the location where you have stored the sevenSegLED.zip file and click on it to open.
- ❏ Verify the file opens and is added as shown in **Figure 16**.
- ❏ You can now add this library to your code by clicking on the library name as in **Figure 17**.
- ❏ Verify that clicking on the *sevenSegLED* library causes the *include* file to be written to the source code as indicated in **Figure 18**.
- ❏ This process is useful when writing a program from scratch. However, we will use source code that already has this *include* file, so delete it.
- ❏ Insert the following program into the Arduino IDE:

```
// A101_ch8_7seg_message_sign Joe Pardue
// 6/10/14

#include "sevenSegLED.h"

String readString;

void setup(){

  Serial.begin(57600);
  Serial.println("A101_ch8_message_sign rev
  1.2");

  setUpLEDs();
}

void loop(){

  String messageString;

  while (Serial.available()) {
    char c = Serial.read();
      //gets one byte from serial buffer
    readString += c;
      // makes the string readString
    delay(2);  // slow to allow buffer to
               // fill with next character
  }

  if(readString.length() > 0)
  {
    messageString = readString;
    Serial.print("You sent messageString: ");
    Serial.println(messageString);
  }

  for (int i = 0; i < readString.length();
  i++)
  {
     setLEDs(readString.charAt(i));
     delay(300);
  }

  readString = "";
}
```

- ❏ Compile the code and then open the serial monitor.
- ❏ This code only prints integers and capital letters, so to test it in the serial monitor enter HELLO WORLD (in all caps) and verify that this message is shown on the LED.
- ❏ Next, enter 1234567890 and verify that the numbers print.

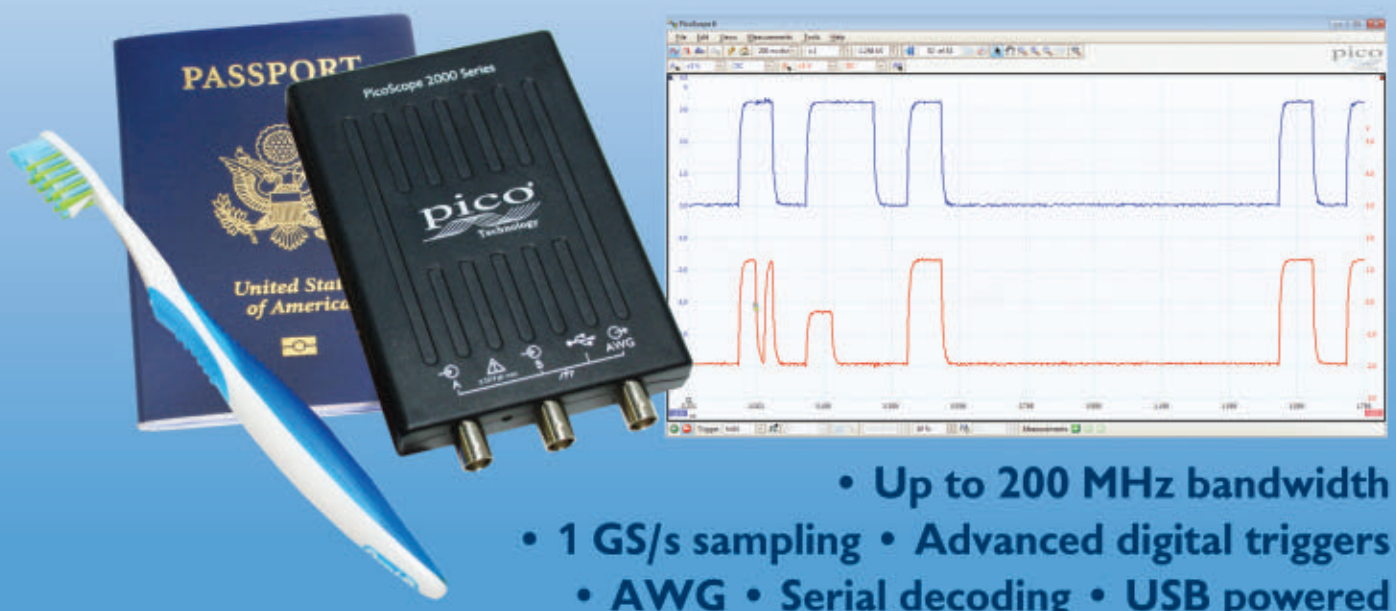
Next month, we'll learn to use light and temperature sensors with the Arduino. Remember that all the components used in the Arduino 101 series are available from the *Nuts&Volts* webstore if you need anything. **NV**

ELECTRONET

pico® Technology

PicoScope® 2200A Series

Like a benchtop oscilloscope, only smaller and better



- Up to 200 MHz bandwidth
- 1 GS/s sampling • Advanced digital triggers
- AWG • Serial decoding • USB powered
- Ultra compact design

PicoScope® 5000 Series

Flexible resolution oscilloscopes

- Resolution from 8 to 16 bits
- 200 MHz analog bandwidth
- 1 GS real-time sampling
- 512 MS buffer memory
- 200 MS/s AWG

PicoScope	PicoScope 5442	PicoScope 5443	PicoScope 5444
Channels	4	4	4
Bandwidth	All modes: 60 MHz	8 to 15-bit modes: 100 MHz 16-bit mode: 60 MHz	8 to 15-bit modes: 200 MHz 16-bit mode: 60 MHz
Sampling rate - real time	1 GS/s (8-bit mode)		
Buffer memory (8-bit) *	32 MS	128 MS	512 MS
Buffer memory (≥ 12-bit)*	16 MS	64 MS	256 MS
Resolution (enhanced)**	8 bits, 12 bits, 14 bits, 15 bits, 16 bits Hardware resolution + 4 bits		
Signal Generator	Function generator or AWG		

2 Channel models also available * Shared between active channels ** Maximum resolution is limited on the lowest voltage ranges: ±10 mV = 8 bits • ±20 mV = 12 bits. All other ranges can use full resolution.

ALL MODELS INCLUDE PROBES, FULL SOFTWARE AND 5 YEAR WARRANTY. SOFTWARE INCLUDES MEASUREMENTS, SPECTRUM ANALYZER, SDK, ADVANCED TRIGGERS, COLOR PERSISTENCE, SERIAL DECODING (CAN, LIN, RS232, I²C, I²S, FLEXRAY, SPI), MASKS, MATH CHANNELS, ALL AS STANDARD, WITH FREE UPDATES.

www.picotech.com/pco525

■ BY LOU FRENZEL W5LEF

White Space Wireless Ready for New Services

Spectrum sharing concepts provide new wireless opportunities.

It is no secret that we are running out of spectrum space for new and/or improved wireless services. The cellular companies are desperate for new spectrum to roll out their Long Term Evolution (LTE) 4G networks and add more subscribers. They pay billions of dollars at government auctions for small chunks of new spectrum, but there is just not much left of the "good" spectrum in the 500 MHz to 4 GHz range to acquire. And that's just for cellular. What about other wireless services? One potential solution for all those other services is white space spectrum.

WHITE SPACE SPECTRUM

White space is unused TV channels that exist across the country. When the Federal Communications Commission (FCC) assigns frequencies for TV stations, they leave gaps in the spectrum (unused channels) to prevent one station from interfering with another in the same area. These empty channels can now be used for other wireless services without a license if specific guidelines are met.

The TV channel spectrum in the US is from 54 MHz (channel 2) to 698 MHz (channel 51). Each channel is 6 MHz wide. Refer to **Table 1** for a complete listing. Channels 3, 4, and 37 are off limits. Channels 3 and 4 are still used for connecting external RF modulators to older analog TV sets. Channel 37 is reserved for the astronomers who use radio telescopes to monitor the universe; these frequencies are ideal for listening to certain types of radiation. In Europe, the white space extends from 54 MHz to 768 MHz with 8 MHz wide channels. The allocation is made in the U.K. by Ofcom – the FCC equivalent.

White space is also called TV white space (TVWS). While channels 2 through 51 are available, the frequency range is way too wide for practical receiver and transmitter circuits. Furthermore, the antenna lengths at the lower frequencies are way too long for portable or mobile devices. Generally speaking, only channels 14 through 51 are practical. So, when you hear the term TVWS, think of channels 14 (470 MHz) to channel 51 (698 MHz).

Incidentally, you will hear the term Super Wi-Fi applied to TVWS. This has nothing to do with the real Wi-Fi wireless local area network (WLAN) standard 802.11. The term implies a longer range than authentic Wi-Fi at 2.4 GHz or 5 GHz whose range is usually restricted to 100 meters or so max.

On the other hand, the lower frequencies of TVWS have signal propagation characteristics that permit much longer ranges of communications. The physics of wireless say that lower frequencies (longer wavelengths) travel farther for a given transmitter power, receiver sensitivity, and antenna gains than do higher frequencies. These lower frequencies also penetrate buildings, trees, and other obstacles better than the microwaves of Wi-Fi or other shorter range wireless like Bluetooth or ZigBee. For some services, TVWS is ideal spectrum.

TVWS APPLICATIONS

The most likely use for TVWS is wireless Internet service in rural areas where there is no cable TV or DSL service. Homes or small businesses will use a wireless modem to talk to a base station connected to the Internet. There are many remote, isolated, or boonies areas in the US without a good Internet connection. This applies to many other developing countries, as well.

TVWS can also be used for wireless backhaul for other services. It could, for example, be a backhaul

Go to **www.nutsvolts.com/index.php?/magazine/article/august2014_OpenComm** to comment on this article.

TV Channel Number	Frequency (MHz)
2	54–60
3	60–66 (not allowed)
4	66–72 (not allowed)
5	76–82
6	82–88
7	174–180
8	180–186
9	186–192
10	192–198
11	198–204
12	204–210
13	210–216
14	470–476
15	476–482
16	482–488
17	488–494
18	494–500
19	500–506
20	506–512
21	512–518
22	518–524
23	524–530
24	530–536
25	536–542
26	542–548
27	548–554
28	554–560
29	560–566
30	566–572
31	572–578
32	578–584
33	584–590
34	590–596
35	596–602
36	602-608
37	608-614 (not allowed)
38	614-620
39	620-626
40	626-632
41	632-638
42	638–644
43	644-650
44	650–656
45	656–662
46	662–668
47	668–674
48	674–680
49	680–686
50	686–692
51	692–698

Table 1 - TV White Space channels and frequencies allowed by the FCC.

for a remote Wi-Fi access point. Another use is backhaul of locally collected smart metering data for utilities.

Most of the newer applications will probably come from the machine-to-machine (M2M) arena where companies are using wireless to monitor and control remote devices or facilities. Some existing applications include pipelines, oil rigs, tank farms, smart grid electrical utility substations, and farm acreage. A good use of TVWS is remote video surveillance. There is no restriction on the use, so it will be interesting to see what new applications will come of TVWS.

GUIDELINES FOR USING TVWS

The FCC approved the use of white space, but the rules and regulations make it tricky and very atypical of other wireless technologies. The whole objective is to prevent interference to TV stations or other services that also use TV channels – like the thousands of wireless microphones. The TVWS radios must be intelligent and have features to limit their operation to channels that will not harm other services. White space radios, therefore, are software-defined cognitive radios that can sense and otherwise determine their location and any surrounding services.

The primary way that a TVWS radio base station knows what is around it is to access a remote database that has recorded all TV stations, wireless mikes, and other stuff for all geographic areas. The base station has GPS capability that determines what segment of the database is accessed via the Internet. The FCC has approved several such comprehensive databases from companies like Spectrum Bridge, iconectiv (previously Telecordia), Google, Microsoft, and others. The database returns local channel assignment information so the radio can find a vacant channel. The radio also listens to sense activity if present. Once a blank or "white space" is identified, the radio sets its frequency and lets any connected remote terminals know so communications can begin. The cognitive software takes all this information and makes a channel selection by tuning the frequency-agile radio with a phase-locked loop (PLL) frequency synthesizer.

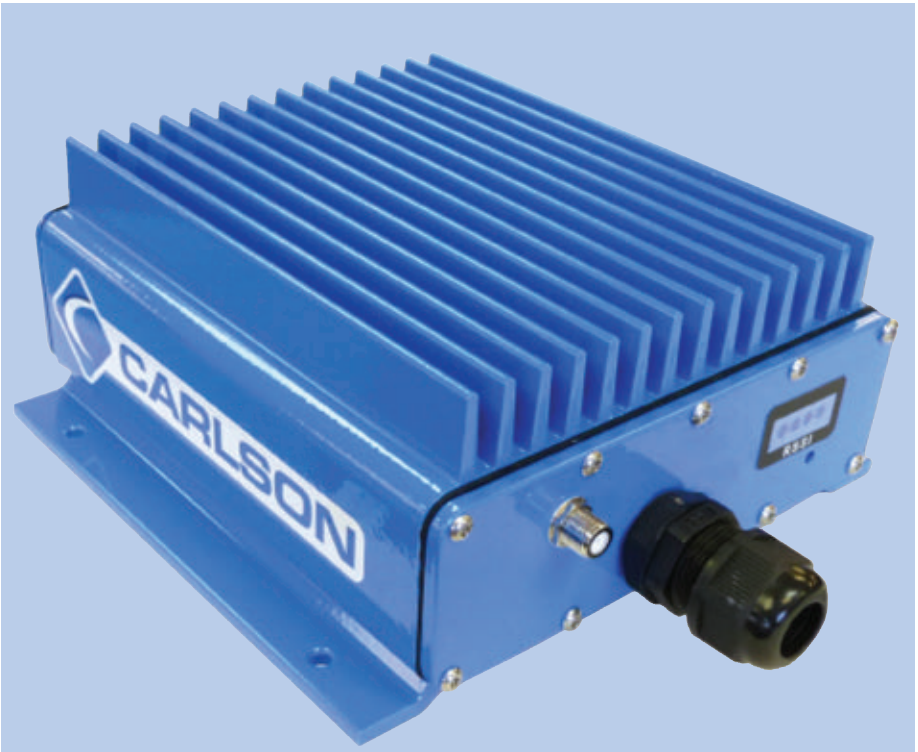
The FCC approved the use of white space, but the rules and regulations make it tricky and very atypical of other wireless technologies. The whole objective is to prevent interference to TV stations or other services that also use TV channels – like the thousands of wireless microphones.

Other features that minimize the potential for interference are power and antenna restrictions. Base station power is limited to one watt or four watts effective isotropic radiated power (EIRP). The EIRP is the transmitted power with a gain antenna. Remote terminals are limited to 40 milliwatts or 100 mW EIRP. Antenna height is limited to 30 meters maximum above ground to restrict range.

RADIO STANDARDS

Most wireless systems conform to fixed standards that meet regulatory requirements, as well as help ensure interoperability between equipment from different manufacturers. So far, no one specific

■ FIGURE 1. The Carlson Wireless RuralConnect white space base station.



■ **FIGURE 2.** A remote user white space radio showing an elevation map and station coverage.

Sources of Additional Information

➤ Code of Federal Regulations (CFR) Title 47, Part 15, and Subpart H.

➤ Dynamic Spectrum Alliance
www.dynamicspectrumalliance.org

➤ Federal Communications Commission
Third Memorandum Opinion and Order (FCC 12-36)
related to Dockets 04-186 and 02-380.
http://hraunfoss.fcc.gov/edocs_public/attachmatch/FFCC-12-36A1.pdf

➤ Industry Internet Consortium
www.iiconsortium.org

➤ Weightless Special Interest Group (SIG)
www.weightless.org

➤ White Space Alliance
www.whitespacealliance.org

standard has been selected. Current vendors use their own proprietary standards. This works well in limited local systems. Nevertheless, the low volume of equipment keeps prices high. One standard would do a lot for higher volume and lower costs. Luckily, several standards are either being developed or are available now.

One of these promising standards is the IEEE 802.11af. This is a version of the popular Wi-Fi standard modified to work in the TVWS bands and meet FCC requirements. It uses the popular OFDM modulation format using BPSK, QPSK, 16QAM, and 64QAM. Maximum potential data rate is 12 Mb/s. It uses time division duplexing (TDD) for two-way communications. For access from multiple stations, the method is carrier sense multiple access with carrier sensing (CSMA/CS) and TDMA. The overall range is several kilometers under the right conditions. The standard is in the final stages of development and it should be finalized by the end of the year.

An older IEEE standard is 802.22 which is ready for use right now. It was developed as an alternative to WiMAX for metro area networks. It is called the wireless regional area network (WRAN) standard. It too uses OFDM with QPSK, 16QAM, or 64QAM modulation. Data rate is up to about 22 Mb/s. It uses OFDMA for multiple access, and duplexing is via TDD. The 802.22 standard also incorporates sophisticated forward error correction (FEC) coding that

really extends reliability, data rate, and range. Maximum range is in the 15 km to 30 km area and up to 100 km under ideal conditions.

An interesting new standard out of the U.K. is called Weightless. It is an attempt to establish a universal worldwide standard for M2M applications in the white space bands. Instead of OFDM, the standard uses single carrier modulation of BPSK, QPSK, and 16QAM. Data rates can be up to 16 Mb/s. It uses TDMA access. The protocol is specifically designed to offer low duty cycle to ensure very low power consumption from remote battery powered modules. It includes strong authentication and encryption for ultra secure data transmissions. Weightless and some related hardware in the form of chips and modules are available now.

■ **FIGURE 3.** The Carlson white space base station antenna.

A TYPICAL SYSTEM

Carlson Wireless is a company that has been making and testing white space equipment for years. Their RuralConnect system has been widely tested in the US and abroad. The base station is shown in **Figure 1** and operates over the 470 MHz to 768 MHz range, covering both US and European bands. Power level is 200 mW and receiver sensitivity is -93 dBm using QPSK. Data rates vary with modulation types of BPSK, QPSK, and 16QAM, and run from 4 Mb/s to 16 Mb/s. Maximum range is up to 12 miles. Access is TDMA and duplexing is TDD. The remote user devices look like **Figure 2** and have similar specifications.

Carlson also offers a great antenna for the base station shown here in **Figure 3**. It is an omnidirectional vertically polarized multi-element array with a gain of 5.2 dBi. It can operate over the whole frequency range from 470 MHz to 768 MHz with low VSWR.

Other equipment makers include Adaptrum, KTS Wireless, Neul Ltd, Redline Communications, and Ruckus with more on the way.

THINGS TO COME

The whole white space phenomenon is still new and developing. It will no doubt find its niche. What is needed is some off-the-shelf ICs for the various standards to simplify, speed up, and lower the cost of new equipment design. In addition, watch for possible spectrum limitations as the FCC plans to auction off spectrum in the 600 MHz to 700 MHz range in the near future. This could limit the number of channels available for white space to channels 14 to 36 instead of up to channel 51. That will not kill white space but it will limit some operations in some locations. We'll have to wait and see. **NV**

 Continued from page 9

down the functional obsolence road too many times.

2. The schematic in Figure 2 shows pins 1, 2, and 3 for IC1b where they should be pins 7, 6, and 5 respectively.

3. The parts list shows D1-D3 for the control board as 1N916. Should they be 1N914s as shown on the schematic?

Looking forward to the frequency counter portion of the build.

Gerry Shand

(Ed note: Check out this issue for the frequency counter!)

Good catch on the IC, Gerry. Since either side can be flip-flopped function wise, I must have labeled IC1a pin numbers and then wandered off to another section of the manuscript. I guess when I came back to that point, I absent-mindedly labeled IC1b with the same pin numbers and never caught my mistake. As to the 1N914/1N916 confusion (and along with 1N4148s), they are all virtually identical. I stock tons of all of these and just grab one when needed without even looking at part numbers, So, I must have been thinking "14s" at one point and "16s" at another point. I can see where it may create some confusion. One more reason why I always advise readers to download datasheets for the semiconductors involved.

As to obsolete semiconductors, only the MC1648 is out of production, but is readily available on eBay in small quantities and low prices. Its new 'baby brother' as indicated in the parts list is in current production. The only other hard to obtain part (the SMV1404-09) is still in current production in the USA. Unfortunately, the aftermarket dried up about the time the manuscript was submitted. I felt a certain responsibility to buy the minimum quantity from the manufacturer (quite expensive) just to supply those interested in building this project with an affordable quantity needed. I did not make one red cent on the sale of these.

This varactor was chosen because it was one of the few that could satisfy all the design requirements and, in my case, has a proven track record over the last 15 years. Now as to availability – even devices that are in current production can at times be hard to obtain as suppliers will have in-stock availability and prices change by the day. This is a trend I am seeing more and more with passing time. EOL (end of life) alerts are happening much sooner than they used to, thanks to obsolescence by the

increasing speed of technological advances. So, nothing is guaranteed or forever.

While we are on the subject of errors, there are a couple I have spotted since the June issue came out. Referring to page 27 and Figure 1, regarding RF band switch S1b as written, it should have been: L1-L8 as per the Parts List instead of L1-L10.

On page 32, second column/ second paragraph, it should have been written as:

Tighten the switch mounting nut. Solder the S1b wiper lead to post "A" and install the wire with FB1 from the ground plate to post "B." Add a wire from the S1a wiper contact to the low end of R12. Now, the individual band coils will be installed.

Robert Reed

Question on Q&A

The Q&A column was diversified and an interesting read. The book — as you know — has been missing this portion for quite some time now. Is it going to return or is that a thing of bygone days? Can you not rotate the column with the contributing editors you have? It`s time for change.

Robert Haskett

Thank you for taking the time to write about your concerns with Nuts & Volts. In regards to the missing Q&A column, long time columnist and faithful contributor Russ Kincaid has hung up his soldering iron and retired from writing the column. We have been actively trying to find someone to take up where Russ left off. The bad news is that Russ had some pretty big shoes to fill.

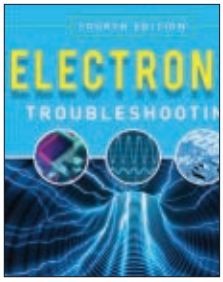
The good news is that just before going to print with this issue, we have confirmed that Tim Brown will be taking over the Q&A column. Many of you will know Tim from his long time participation in the Tech Forum. Tim has a history of detailed, well thought out responses covering a wide range of topics, as well as a knack for writing in an easy to understand style. We hope now to have the Q&A column back up by the October — or at the latest — November issue. So, start sending in those tech questions to Q&A@nutsvolts.com.

Also, we are working with a new bunch of writers, in addition to your old favorites! These folks should add some new flavor to the magazine and be a welcome change. Look for some of these new writers to appear in the September issue which should be a pretty big departure from "magazine as usual" around here!

Vern Graner
VP of Operations

GREAT FOR DIYers!

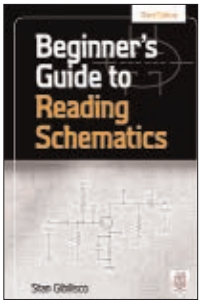
Electronic Troubleshooting, Fourth Edition
by Daniel Tomal, Aram Agajanian



Electronic Troubleshooting, Fourth Edition provides technicians with a wealth of problem-solving methods and information on troubleshooting theory, techniques, and practices for a wide variety of electrical and electronic devices. Special emphasis is placed on the digital electronics and microprocessor-based systems that are used in today's industrial and personal applications.

Regular Price $70.00
Sale Price $59.95

Beginner's Guide to Reading Schematics, 3E
by Stan Gibilisco



Navigate the roadmaps of simple electronic circuits and complex systems with help from an experienced engineer. With all-new art and demo circuits you can build, this hands-on, illustrated guide explains how to understand and create high-precision electronics diagrams. Find out how to identify parts and connections, decipher element ratings, and apply diagram-based information in your own projects.

$25.00

Make Your Own PCBs with EAGLE
by Eric Kleinert

Featuring detailed illustrations and step-by-step instructions, *Make Your Own PCBs with EAGLE* leads you through the process of designing a schematic and transforming it into a PCB layout. You'll then move on to fabrication via the generation of standard Gerber files for submission to a PCB manufacturing service. This practical guide offers an accessible, logical way to learn EAGLE and start producing PCBs as quickly as possible.

$30.00

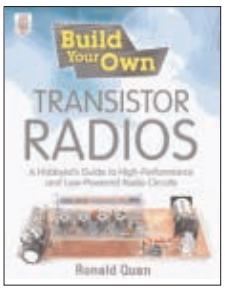
Programming the BeagleBone Black: Getting Started with JavaScript and BoneScript
by Simon Monk

Learn how to program the BeagleBone Black — the wildly popular single-board computer — using JavaScript and the native BoneScript language. You'll find out how to interface with expansion capes to add capabilities to the basic board, and how to create a Web interface for BBB. Two hardware projects demonstrate how to use the board as an embedded platform.

$15.00

Build Your Own Transistor Radios
by Ronald Quan

A Hobbyist's Guide to High Performance and Low-Powered Radio Circuits



Create sophisticated transistor radios that are inexpensive yet highly efficient. Inside this book, it offers complete projects with detailed schematics and insights on how the radios were designed. Learn how to choose components, construct the different types of radios, and troubleshoot your work.

***Paperback, 496 pages**
$49.95

The Steampunk Adventurer's Guide
by Thomas Willeford

Steampunk stalwart Thomas Willeford cordially invites you on an adventure — one in which you get to build ingenious devices of your own! Lavishly illustrated by award-winning cartoonist Phil Foglio, *The Steampunk Adventurer's Guide: Contraptions, Creations, and Curiosities Anyone Can Make* presents 10 intriguing projects ideal for makers of all ages and skill levels, woven into an epic tale of mystery and pursuit.

$25.00

How to Diagnose and Fix Everything Electronic
by Michael Jay Geier

Master the Art of Electronics Repair

In this hands-on guide, a lifelong electronics repair guru shares his tested techniques and invaluable insights. *How to Diagnose and Fix Everything Electronic* shows you how to repair and extend the life of all kinds of solid-state devices, from modern digital gadgetry to cherished analog products of yesteryear.

$24.95

Programming PICs in Basic
by Chuck Hellebuyck

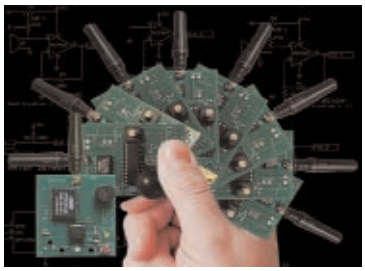
If you wanted to learn how to program microcontrollers, then you've found the right book! Microchip PIC microcontrollers are being designed into electronics throughout the world and none is more popular than the eight-pin version. Now the home hobbyist can create projects with these little microcontrollers using a low cost development tool called the CHIPAXE system and the Basic software language. Chuck Hellebuyck introduces how to use this development setup to build useful projects with an eight-pin PIC12F683 microcontroller. **$14.95**

Programming Arduino Next Steps: Going Further with Sketches
by Simon Monk

In this practical guide, electronics guru Simon Monk takes you under the hood of Arduino and reveals professional programming secrets. Also shows you how to use interrupts, manage memory, program for the Internet, maximize serial communications, perform digital signal processing, and much more. All of the 75+ example sketches featured in the book are available for download. **$20.00**

PROJECTS

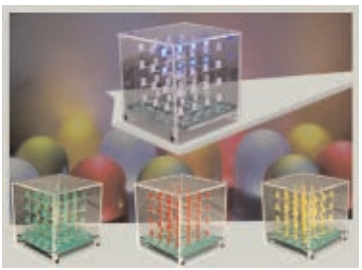
Super Detector Circuit Set



Pick a circuit!
With one PCB you have the option of detecting wirelessly: temperature, vibration, light, sound, motion, normally open switch, normally closed switch, any varying resistor input, voltage input, mA input, and tilt, just to name a few.

$32.95

3D LED Cube Kit



This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.
Colors available: Green, Red, Yellow & Blue

$57.95

Solar Charge Controller Kit 2.0

Now with more options!
If you charge batteries using solar panels, then you can't afford not to have them protected from over-charging. This 12 volt/12 amp charge controller is great protection for the money. It is simple to build, ideal for the novice, and no special tools are needed other than a soldering iron and a 9/64" drill!

$27.95

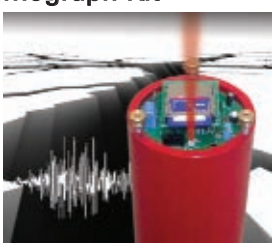
Geiger Counter Kit

As seen in the March 2013 issue.
This kit is a great project for high school and university students. The unit detects and displays levels of radiation, and can detect and display dosage levels as low as one micro-roentgen/hr. The LND712 tube in our kit is capable of measuring alpha, beta, and gamma particles.

Partial kits also available.

$145.95

Seismograph Kit



As seen in the May 2012 issue.
Now you can record your own shaking, rattling, and rolling.
The Poor Man's Seismograph is a great project /device to record any movement in an area where you normally shouldn't have any. The kit includes everything needed to build the seismograph. All you need is your PC, SD card, and to download the free software to view the seismic event graph.

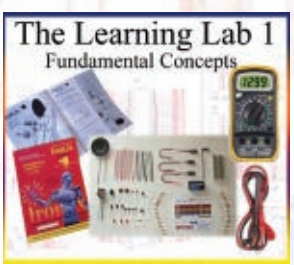
$79.95

Battery Marvel

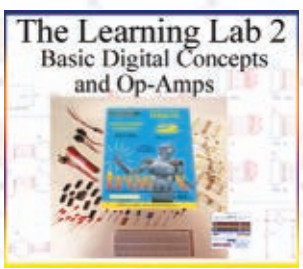
As seen in the November 2011 issue.
Battery Marvel helps protect cars, trucks, motorcycles, boats, and any other 12V vehicles from sudden battery failure. This assembled unit features a single LED that glows green, yellow, or red, indicating battery health at a glance. An extra-loud piezo driver alerts you to any problems.
Details, please visit our website.

$32.95

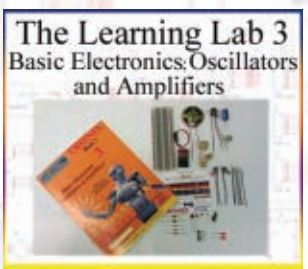
FOR BEGINNER GEEKS!



$59.95



$49.95



$39.95

The labs in this series — from GSS Tech Ed — show simple and interesting experiments and lessons, all done on a solderless circuit board.
As you do each experiment, you learn how basic components work in a circuit, and continue to build your arsenal of knowledge with each successive experiment.

For more info and a promotional video, please visit our webstore.

CLASSIFIEDS

SURPLUS

SURPLUS ELECTRONIC PARTS & ACCESSORIES

Over 20,000 Items in Stock.

Cables, Connectors, Displays, Fans, Hardware, LEDs, Motors, Potentiometers, Relays, Semiconductors, Service Manuals, Speakers, Switches, Test Equipment, Tools, VCR Parts

Surplus Material Components
SMC ELECTRONICS
www.smcelectronics.com

No Minimum Order.
Credit Cards and PAYPAL Accepted.
Flat $4.95 per order USA Shipping.

LIGHTING

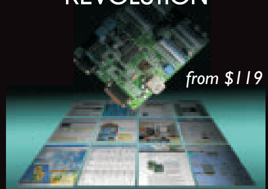
- LED modules
- Solar controls
- Dusk to dawn
- Dimmers12/24V

J2 LED LIGHTING LLC
www.j2ledlighting.com

Subscribe today!
www.nutsvolts.com

CONTROLLERS

Join The INTERNET of THINGS REVOLUTION



TRI SUPER PLCs

Powerful & Easy Ladder
+BASIC Programming
Ethernet integrated
MODBUS TCP/IP
DI/Os & AI/Os integrated

tel : 1 877 TRI-PLCS
web : www.triplc.com/nv.htm

TRIANGLE RESEARCH INTERNATIONAL

HARDWARE WANTED

DEC EQUIPMENT WANTED!!!

Digital Equipment Corp. and compatibles.
Buy - Sell - Trade

CALL KEYWAYS 937-847-2300
or **email buyer@keyways.com**

KITS/PLANS

QKITS LTD
sales@qkits.com
1 888 GO 4 KITS

Arduino • Raspberry Pi
Power Supplies
MG Chemicals
RFID

Visit us at:
www.qkits.com

ROBOTICS

MaxBotix®
High Performance Ultrasonic Rangefinders

Check us out Today!

Save 10% NOW On Web Orders
Enter Code: NV311408

HRXL-MaxSonar®-WR™
- High noise tolerance
- IP67 rated
- 1 mm resolution
- Multi-Sensor operation
- Calibrated beam pattern
- Starting at $109.95

XL-MaxSonar®-EZ™
- Great for UAV's and robotics
- Incredible noise immunity
- Small in size
- 1cm resolution
- Automatic calibration
- Starting at $39.95

Phone: 218-454-0766 Email: sales@maxbotix.com
www.maxbotix.com

www.servo magazine.com

ELECT DATA SYSTEMS

S6000:TTL to Ethernet Converter

$17.99

- Au□o de□ec□10/100M □□gh □□eed □□her□e□
- □aud ra□e 300 □25600 □□□
- Agreeme□□: □T□□□□□T□A□P□□P□□□P□T□P
- □a□□or□a□□□r□ua□□□M
- 3.3□ or 5□ □o□er □□□u□□□er□ace

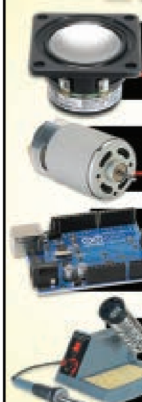
□□her□e□□□g□ee□□□485 □□ modu□e□a□
WWW.SHJELECTRONIC.COM

AUDIO/VIDEO PARTS

ICEpower MODULES
50W x 2 (50ASX2) or
170W x 1 (50ASX2BTL)
Just add AC & speaker(s)
$99 USA $119 Worldwide
www.classicaudioparts.com

PARTS EXPRESS
YOUR ELECTRONICS CONNECTION™

OVER 18,000 ELECTRONIC PARTS IN STOCK



Call or visit us online today to receive your **FREE** copy of our 2014 catalog!

parts-express.com/nuts
1-800-338-0531

MISC FOR SALE

SPEAKER REBUILDING BUSINESS CLOSED AFTER 35 YEARS

ISE IS LIQUIDATING $250,000 INVENTORY OF SPEAKERS, VOICE COILS, CONES, MAGINETS, GASKETS, SPIDERS, MANUFACTURING EQUIPMENT.
ALL OFFERS CONSIDERED.
CALL ISE 956-444-0004, 888-351-5550
www.iseliquidator.com

SERVICES

Sandtronics

Sandpiper Electronics Services LLC
"design of custom electronics instrumentation"

Circuit design & drawings. PCB layout, fabrication & population
Prototype development for research, industry and inventors.
39 years experience in electronics development for research & flight applications , laser systems & laboratory research

FREE no obligation consultations
www.sandtronics.com

Did You Know ...

Preferred Subscribers get access to all digital back issues of *Nuts & Volts* for free?

Call for details
1-877-525-2539
or go to
www.nutsvolts.com/faqs

READER-TO-READER TECHFORUM

All New TECHFORUM Online At www.nutsvolts.com/tech-forum

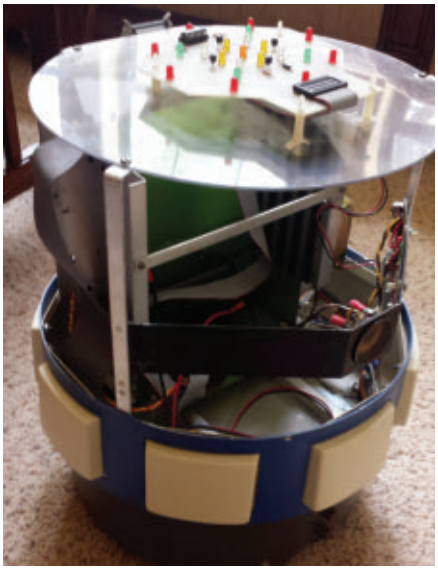
>>> QUESTIONS

Audio Mixer Question

I have a rather old audio mixer that has 1/4" inputs for microphones. The inputs are labeled "HiZ." I have microphones I'd like to use, but they have a three-pin "XLR" style connector. Do I need a some kind of matching transformer, or can I just use a 1/4" male plug wired to some of the contacts on the XLR jack?

#8141 **Andre La Tores**
Birmingham, AL

Help Identifying an Old Robot



I picked up an old robot chassis at an estate sale for $25. I have taken some pictures in the hope someone might be able to identify the make/model for me.

#8142 **Jaime Smithers**
Las Vegas, NV

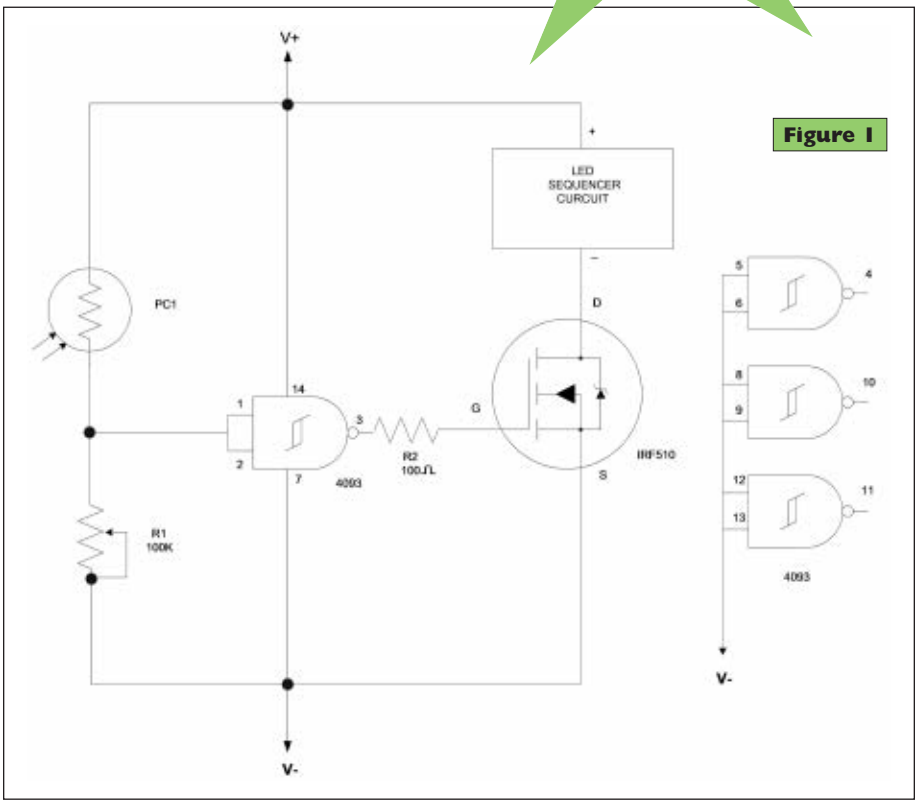


Figure 1

iPod Battery Reconditioning

My son has gifted me with his cast-off iPod Classic 30gb player. Unfortunately, the battery life is only about 15-20 minutes of playing before it goes dead. Does anyone know of a way to recondition the battery?

#8143 **Wendal Haynes**
Memphis, TN

Flashlight to Flashing Light

I have a small LED flashlight that I mount on my handlebars while riding my bike at night. I've seen some people put a flashing white light on the front of their bikes. Is there a circuit to convert my "regular" flashlight to a "flashing" flashlight?

#8144 **Sandy McCain**
Fullerton, CA

>>> ANSWERS

[#7141 - July 2014]

Photoresister Switcher

I would like to know if I have correctly connected the photoresistors (CdS photocells) shown here in order to turn OFF during the day and to turn ON during the night the two LED circuits attached to them. (If not, please indicate by a new diagram.)

Send all questions and answers by email to **forum@nutsvolts.com** or via the online form at **www.nutsvolts.com/tech-forum**

All questions *AND* answers are submitted by *Nuts & Volts* readers and are intended to promote the exchange of ideas and provide assistance for solving technical problems. All submissions are subject to editing and will be published on a space available basis if deemed suitable by the publisher. Answers are submitted by readers and **NO GUARANTEES WHATSOEVER** are made by the publisher. The implementation of any answer printed in this column may require varying degrees of technical experience and should only be attempted by qualified individuals.

Always use common sense and good judgment!

Send all questions and answers by email to **forum@nutsvolts.com**
or via the online form at **www.nutsvolts.com/tech-forum**

Also, I would like to know:

a) *If any photoresistor would work?*

b) *What would the optimum dark/light resistance values be for such a photoresistor?*

c) *How would I calculate the values (any formula?) from the transistor side (2N2222) that would best fit this ON/OFF photoresistor switcher?*

The two circuits Nate submitted are configured to turn the LED sequencers ON during the day and OFF at night. This is because the resistance of the photocell decreases as light falling on the photocell increases, causing increased current flow from the base of the transistor to the positive power supply rail. The increase in base current will cause the transistor to conduct and supply current to the LED sequencer circuits in daylight. To modify the circuits so they turn OFF during the day and ON at night, swap the photocell and the resistor so that decreasing photocell resistance will pull the base of the transistor toward ground, turning the transistor off in the daytime. The fixed resistor will turn the transistor on at night when the resistance of the photocell is high compared to the resistor. The circuit can be improved *(*see new diagram in **Figure 1***)*. Two issues are solved by the improved circuit.

1) Nate is using a 2N2222 transistor as a low side switch to control current to the light sequencer circuits. A MOSFET makes a better switch because it has a very high OFF resistance and a very low ON resistance, so more of the supply voltage is applied to the light sequencer circuit and less power is wasted in the transistor. For this reason, I replaced the 2N2222 transistor with a low cost IRF510 N-channel power MOSFET.

2) The other problem with Nate's circuits is they won't turn on and off quickly like a mechanical switch. At dawn and dusk — when the light changes gradually — the current through the transistor will also change gradually. The switching action should happen fast when a predetermined light threshold is crossed; a Schmitt trigger circuit is needed for that. The improved circuit contains four Schmitt trigger NAND gates in a single 4093 CMOS IC. Only one NAND gate is needed, so the other inputs should be tied to V- to prevent instability.

The 4093 has a hysteresis band which keeps the sequencer circuit from receiving rapid bursts of current at dawn and dusk when the control voltage hovers near the tripping point. The output of the 4093 drives the IRF510 MOSFET which switches the current to the LED light sequencer circuit on and off. The MOSFET is driven into saturation by the 4093 Schmitt trigger to provide clean switching and maximum current to the LED sequencer circuit. Next, the answers to questions a, b, and c.

a) *Will any photoresistor work?* Most photocells will work with the improved circuit, but don't confuse a photocell (which is a photoresistor) with a phototransistor. I've included a pot wired as a variable resistor; this will adjust the voltage divider to match the photocell characteristics. To set the pot, go outside in the twilight near sunset and adjust the pot until the circuit turns ON, then back off slowly until it turns off. The LED sequencer circuit should then turn on at sunset and off at sunrise.

b) *What are the optimum photoresistor light/dark resistance values?* The resistance of most photocells varies from a few hundred ohms (or less) in direct sunlight to a megohm (or more) in total darkness. The optimum resistance of R1 for a given photocell can be calculated using Ohm's Law, but the adjustment pot in the improved circuit makes it compatible with most CdS photocells.

c) *What formula will calculate the optimum component values?* The 2N2222 transistor used in Nate's circuits is a current operated device which should be explained in terms of current. The improved circuit uses voltage controlled components. *This explanation is only for the improved circuit.*

Inputs 1 and 2 of the 4049 are tied together and connected to a voltage divider consisting of photocell PC1 and resistor R1, which are connected in series between V+ and V-. To predict circuit operation in different lighting conditions, calculate the voltage from V- to the point where R1 and PC1 join using Ohm's Law, and then compare this voltage to the upper and lower trip points of the 4093. The lower trip point will turn the LED sequencer circuit on because the 4093 is an inverter, which will drive the gate of the MOSFET to V+ when the 4093 input is low. When the MOSFET conducts, it will ground the LED sequencer circuit to V-. The lower trip point of the 4093 is +3.9 volts when V+ is 10 volts and the upper trip point is +5.9 volts.

EVENING CONTROL: The voltage across R1 must fall below 3.9 volts to turn the LED sequencer on at sunset; this example uses 3.5 volts to be well below the threshold. To calculate the optimum resistance of R1, it's necessary to know the resistance of PC1 in the twilight hours of sunset or dawn.

This example assumes the resistance of PC1 is 100K at sunset. The voltage drop across both resistors must add up to V+. So, if V+ is 10 volts, then 6.5 volts must be dropped across 100K (PC1) for 3.5 volts to be dropped across R1. Operation of the improved circuit is explained in four steps.

STEP 1: *Calculate the current through the voltage divider at sunset.* If we know the resistance of PC1 and the voltage across it, then we can calculate the current through PC1 with Ohm's Law (I = E/R). Plugging in the known values, I = 6.5/100,000 = 65 µA. This is a series circuit, so 65 µA

Send all questions and answers by email to **forum@nutsvolts.com**
or via the online form at **www.nutsvolts.com/tech-forum**

also passes through R1.

STEP 2: *Calculate the resistance of R1.* We know the current through R1 is 65 µA and the voltage across R1 is 3.5 volts, so we can use another form of Ohm's Law (R = E/I) to calculate the resistance of R1. R1 = 3.5/.000065 = 53846 ohms. To check this, use E = IR, E = .000065 (53846) = 3.499 volts; 3.5 volts is under the lower trip point of the 4093, so output pin 3 of the NAND gate will go high, driving the gate of the IRF510 MOSFET to V+ through R2. This causes the MOSFET to conduct and provide a current path from V- to the LED sequencer circuit.

MORNING CONTROL: At the soft light near daybreak, a typical photocell has a resistance around 10K. If we use the value of 53.846K previously calculated for R1 and assume a PCI resistance of 10K at dawn, then the total resistance of PC1 + R1 is 10K + 53.846K = 63.846K.

STEP 3: *Find the daytime current through the voltage divider.* If V+ is 10 volts, the current through the series resistances PC1 and R1 at sunrise can be calculated with I = E/R. I = 10/63846 = .0001566 A, or 157 µA.

STEP 4: *Find the daytime voltage drop across R1.* Now that we know the morning current through R1 and the resistance of R1, we can calculate the morning voltage drop across R1 with E = I/R; E = .0001566 (53846) = 8.43. A daytime voltage drop across R1 of 8.43 volts is well above the 5.9 volt upper trip point of the 4093, so the output of the inverting 4093 will go low, grounding the gate of the MOSFET to V- and turning off current to the LED sequencer circuit. During the day, the input to the 4093 should vary between 8 and 10 volts, keeping the LED sequencer circuit turned off.

Ed Gore
Panama City, FL

THE HALLOWEEN SPECIAL EDITION IS COMING!

The fall season is just around the corner and to help get everyone in the spirit, we're devoting the September issue of *Nuts & Volts* to spooky projects and fun Halloween-themed articles! Even if you don't have a blood lust for Halloween fun, these articles provide solid examples of servo control systems, sensor operation and circuit crafting that any electronics enthusiast will enjoy.

Not only have we managed to dig up some great new authors, we've also challenged the authors you know by heart with creating inspirational and informative items for your Halloween fun!

Some examples of the articles scheduled to appear include:

SYNCHRONIZED SCARY LAUGHING PUMPKINS

VIDEO PROJECTION GHOSTS AND GHOULS

THE PEEK-A-BOOOOO GHOST

THE GHOST PHONE

PICAXE HAUNT-CONTROL BOARDS

TRIO DE LOS MUERTOS

MONSTER IN A BOX- REVEALED

BUT WAIT! THERE'S MORE!

More?! *Yes MORE!* Simulated flame systems, Talking Skull, Growling monster books, "Haunting 101," the Prop-Dropper 2.0, and other fun and inspiring articles are all scheduled to appear in this all-new action-packed HALLOWEEN SPECIAL EDITION!

THEY'RE NOT DEAD!

Don't Despair! The regularly scheduled columns we all know and love will be back in the October issue of *Nuts & Volts*, so stay tuned!

WWW.NUTSVOLTS.COM

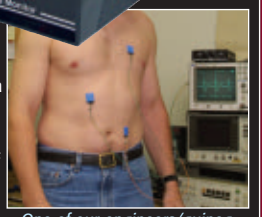
Beat The Heat...Build a Kit!

Ramsey Kits Are Always Neat, Even In The Dog Day Heat!

Electrocardiogram ECG Heart Monitor

- ✔ **Visible and audible display of your heart rhythm!**
- ✔ **Bright LED "Beat" indicator for easy viewing!**
- ✔ **Re-usable hospital grade sensors included!**
- ✔ **Monitor output for professional scope display**
- ✔ **Simple and safe 9V battery operation**

When we think summer, we normally think of vacations, traveling, and all the activities you have been waiting for all winter! And whether that includes hiking that new trail you've heard about or simply riding the new rides (and waiting in line in the scorching heat!) at Wally World, there WILL be physical exertion involved! While we are frequently reminded that February is national Heart Smart month, we think every month should be Heart Smart month. Heart Smart is a way of life, and certainly shouldn't be limited to one month a year. We kept that in mind when we designed the ECG1!

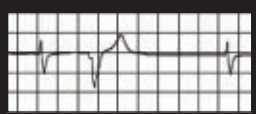


One of our engineers/guinea pigs, checking his heart!

Not only will building an actual ECG be a thrill, but you'll get hands-on knowledge of the relationship between electrical activity and the human body. Each time the human heart beats, the heart muscle causes small electrical changes across your skin. By monitoring and amplifying these changes, the ECG1C detects the heartbeat and allows you to accurately display it, and hear it, giving you a window into the inner workings of the human heart and body!

Use the ECG1C to astound your physician with your knowledge of ECG/EKG systems. Enjoy learning about the inner workings of the heart while, at the same time, covering the stage-by-stage electronic circuit theory used in the kit to monitor it. The three probe wire pick-ups allow for easy application and experimentation without the cumbersome harness normally associated with ECG monitors.

The documentation with the ECG1C covers everything from the circuit description of the kit to the circuit description of the heart! Multiple "beat" indicators include a bright front panel LED that flashes with the actions of the heart along with an adjustable level audio speaker output that supports both mono and stereo hook-ups. In addition, a monitor output is provided to connect to any standard oscilloscope to view the traditional style ECG/EKG waveforms just like you see in a real ER or on one of the medical TV shows!



Look what I found!

On a personal note... See the display to the left? That's me! In between writing this montly ad copy, catalog copy, and plethora of other tasks here, I noticed some skipped beats in my pulse! An immediate cardiac check found I had something called Trigeminy, or PVCs that occur at intervals of 2 normal beats to one PVC! And I saw it with our ECG1 kit!

The fully adjustable gain control on the front panel allows the user to custom tune the differential signal picked up by the probes giving you a perfect reading and display every time! 10 hospital grade re-usable probe patches are included together with the matching custom case set shown. Additional patches are available in 10-packs. Operates on a standard 9VDC battery (not included) for safe and simple operation. Note, while the ECG1C professionally monitors and displays your heart rhythms and functions, it is intended for hobbyist usage only. If you experience any cardiac symptoms, seek proper medical help immediately!

ECG1C	**Electrocardiogram Heart Monitor Kit With Case & Patches**	**$44.95**
ECG1WT	**Electrocardiogram Heart Monitor, Factory Assembled & Tested**	**$89.95**
ECGP10	**Electrocardiogram Re-Usable Probe Patches, 10-Pack**	**$4.95**

12VDC Regulated

Go green with our new 12VDC 1A regulated supply. Worldwide input 100-240VAC with a Level-V efficiency! It gets even better, includes DUAL ferrite cores

AC121 **12VDC 1A Regulated Supply** **$9.95**

12VDC Worldwide

It gets even better than our AC121 to the left! Now, take the regulated Level-V green supply, bump the current up to 1.25A, and include multiple blades for global country compatibility!

PS29 **12VDC 1.25A Global Supply** **$19.95**

Passive Aircraft Monitor

PATENTED!

The hit of the decade! Our patented receiver hears the entire aircraft band without any tuning! Passive design has no LO, therefore can be used on board aircraft! Perfect for airshows, hears the active traffic as it happens! Available kit or factory assembled.

ABM1 **Passive Aircraft Receiver Kit** **$89.95**

Signal Magnet Antenna

The impossible AM radio antenna that pulls in the stations and removes the noise, interference, and static crashes from your radio! Also helps that pesky HD AM Radio stay locked! Also available factory assembled.

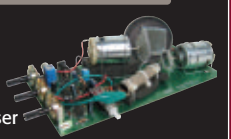
SM100 **Signal Magnet Antenna Kit** **$89.95**

Laser Trip Sensor Alarm

True laser protects over 500 yards! At last within the reach of the hobbyist, this neat kit uses a standard laser pointer (included) to provide both audible and visual alert of a broken path. 5A relay makes it simple to interface! Breakaway board to separate sections.

LTS1 **Laser Trip Sensor Alarm Kit** **$29.95**

Laser Light Show



Just like the big concerts, you can impress your friends with your own laser light show! Audio input modulates the laser display to your favorite music! Adjustable pattern & speed. Runs on 6-12VDC.

LLS1 **Laser Light Show Kit** **$49.95**

The Learning Center!

Beginners To Advanced... *It's Fun!*

- ✔ **Learn, build, and enjoy!**
- ✔ **130, 200, 300, & 500 in one electronic labs!**
- ✔ **Practical through hole and SMT soldering labs!**
- ✔ **Integrated circuit AM/FM radio lab!**
- ✔ **AM, AM/FM, and SWL radio labs!**
- ✔ **Radio Controlled (RC) car!**
- ✔ **Beginner's non-soldering kits!**

For over 4 decades we've become famous for making electronics fun, while at the same time making it a great learning experience. As technology has changed over these years, we have continued that goal!

PL130A Gives you 130 different electronic projects together with a comprehensive learning manual describing the theory behind all the projects.

PL200 Includes 200 very creative fun projects and includes a neat interactive front panel with 2 controls, speaker, LED display and a meter.

PL300 Jump up to 300 separate projects that start walking you through the learning phase of digital electronics.

PL500 The ultimate electronics lab that includes 500 separate projects that cover it all, from the basics all the way to digital programming.

SP1A Whether young or old, there's always a need to hone your soldering skills. Either learn from scratch or consider it a refresher, and end up with a neat little project when you're done!

SM200K Move up to Surface Mount Technology (SMT) soldering, and learn exactly how to solder those tiny little components to a board!

AMFM108K We not only take you through AM and FM radio theory but we guide you through IC's. When you're done you've built yourself an IC based AM/FM radio that works great!

AM2 Learn the complete theory of AM broadcast radio and end up with a highly sensitive AM radio receiver!

AMFM7 Step up to AM/FM with this multi-band lab and learn the basics of both bands. The FM tuner is factory assembled and aligned!

SR3 Enter the world of SWL with this shortwave radio learning lab covering 6-8MHz and 12-18MHz!

AK870 One of the most exciting electronic learning kits that the kids will love! Build a complete RC speedster from the ground up! 7 remote functions!

PL130A	**130-In-One Lab Kit**	**$39.95**
PL200	**200-In-One Lab Kit**	**$84.95**
PL300	**300-In-One Lab Kit**	**$109.95**
PL500	**500-In-One Lab Kit**	**$249.95**
SP1A	**Through Hole Soldering Lab**	**$9.95**
SM200K	**SMT Practical Soldering Lab**	**$22.95**
AMFM108K	**AM/FM IC Lab Kit & Course**	**$36.95**
AM2	**AM Radio Learning Lab**	**$11.95**
AMFM7	**AM/FM Radio Learning Lab**	**$12.95**
SR3	**Short Wave Radio Learning Lab**	**$16.95**
AK870	**RC Speedster Car Kit**	**$29.95**

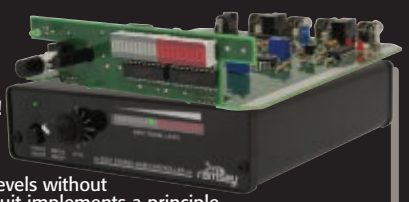
ramsey
www.ramseykits.com

There's only so much room on these two pages, so check it all out in our new virtual electronic catalog! Flip through the pages and search with ease! Visit ***www.ramseycatalog.com***

Follow Us and SAVE $$

Follow us on your favorite network site and look for a lot of super deals posted frequently... exclusively for our followers!

Stereo Audio Platform Gain Controller



- ✔ **Stereo audio processing while preserving audio dynamics!**
- ✔ **True stereo control keeps virtual sonic source location intact!**
- ✔ **Auto-bypass restores original levels when power is turned off!**
- ✔ **Built-in bar graph indication of signal level with display mute!**

The SGC1 is one of our latest kits, and provides a great solution to the age-old problem: how can we easily correct inconsistent audio levels without negatively affecting the dynamics of the audio signal? The SGC1 circuit implements a principle known as the "Platform Gain Principle," which was originally developed by CBS Labs to allow transmitted audio levels to be automatically adjusted to keep them within a desired range.

Think of it like an audio engineer, constantly adjusting the output level in order to limit highs that would be too loud while boosting lower levels so that they can still be heard. You may think "oh, this is just another limiter/compressor!" Not so! Here's the real trick: keeping the full dynamic range ratio of the output signal the same as the original input - something the typical limiter/compressor can only dream of doing! The SGC1 can be placed in just about any standard analog stereo line level audio circuit to keep the audio level within the desired range. It's also the perfect addition to any of our hobby kit transmitters, allowing you to match levels between different audio sources while keeping lows audible and preventing the highs from overdriving.

In addition to its useful basic function and great audio performance, the SGC1 also boasts a front panel LED meter to give an indication of the relative level of the input signal, as well as a front level control that allows you to adjust the controller to the min/max center point of your desired level range. And yes, it is a ***Stereo Gain Controller!*** Meaning that the levels of both the left and right channels are monitored and adjusted equally, thereby maintaining the relative virtual position of things like instruments, singers and speakers! The entire unit is housed in a slim attractive black textured aluminum case that is sure to complement your studio or home theatre. If you're looking for perfect audio levels, hire a broadcast audio engineer, but if that doesn't fit your budget, the SGC1 is the next best thing! Includes 15VDC world-wide power adapter.

SGC1 **Stereo Audio Platform Gain Controller Kit** **$179.95**

IN-8-2 or IN-14 Tubes!

Classic Nixie Tube Clocks

Our next generation of classic Nixie tube clocks perfectly mesh today's technology with the Nixie era technology of the 60's. Of course, features you'd expect with a typical clock are all supported with the Nixie clock... and a whole lot more! The clocks are programmable for 12 or 24 hour mode, various AM/PM indications, programmable leading zero blanking, and include a programmable alarm with snooze as well as date display, 4 or 6 tube, kit or assembled!

We then jumped the technological time line of the 60's Nixie displays by adding the latest multi-colored LEDs to the base of the Nixie tubes to provide hundreds of illumination colors to highlight the glass tubes! The LED lighting can be programmed to any color and brightness combination of the colors red, green, or blue to suit your mood or environment. Then we leaped over the technological time line by integrating an optional GPS time base reference for the ultimate in clock accuracy! The small optional GPS receiver module is factory assembled and tested, and plugs directly into the back of the clock to give your Nixie clock accuracy you could only dream of! The clocks are available in our signature hand rubbed Teak & Maple or clear acrylic bases.

NIXIE **Classic Nixie Tube Clock Kits** **From $229.95**

RF Preamplifier

The famous RF preamp that's been written up in the radio & electronics magazines! This super broadband preamp covers 100 KHz to 1000 MHz! Unconditionally stable gain is greater than 16dB while noise is less than 4dB! 50-75 ohm input. Runs on 12-15 VDC.

SA7 **RF Preamp Kit** **$16.95**

Touch Switch

Touch on, touch off, or momentary touch hold, it's your choice with this little kit! Uses CMOS technology. Actually includes TWO totally separate touch circuits on the board! Drives any low voltage load up to 100mA. Runs on 6-12 VDC.

TS1 **Touch Switch Kit** **$9.95**

Mad Blaster Warble Alarm

If you need to simply get attention, the "Mad Blaster" is the answer, producing a LOUD ear shattering raucous racket! Super for car and home alarms as well. Drives any speaker. Runs on 9-12VDC.

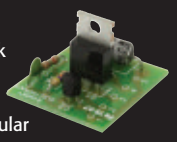
MB1 **Mad Blaster Warble Alarm Kit** **$9.95**

Optically Isolated Module

The hobbyist's headache solver! Converts any AC or DC signal to logic level. The beauty is that the input and output are totally isolated from each other! Output can drive up to 150mA at 40VDC.

OM2 **Optically Isolated Module Kit** **$16.95**

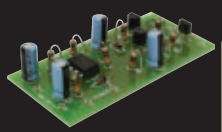
Electronic Siren



What an attention getter! Just hook up a speaker, battery and a switch. Close the switch for an upward wail and release it for the downward return. One of our most popular minikits! Runs on 9V battery.

SM3 **Electronic Siren Kit** **$7.95**

Automatic VOX



This popular VOX switch provides a switched output when it "hears" sound! Use it to key a transmitter, turn on lights, or anything... all controlled by your voice!

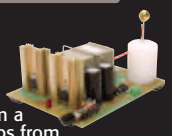
VS1 **Automatic VOX Switch Kit** **$9.95**

Electronic Watch Dog

A barking dog on a PC board! And you don't have to feed it! Generates 2 different selectable barking dog sounds. Plus a built-in mic senses noise and can be set to bark when it hears it! Adjustable sensitivity... unlike my Greyhound and Boxer! 9-12VDC.

K2655 **Electronic Watch Dog Kit** **$39.95**

HV Plasma Generator



Generate 2" sparks to a handheld screwdriver! Light fluorescent tubes without wires! This plasma generator creates up to 25kV at 20kHz from a solid state circuit! Build plasma bulbs from regular bulbs and more! Runs on 16VAC or 5-24VDC.

PG13 **HV Plasma Generator Kit** **$64.95**

Air Blasting Ion Generator

Generates negative ions along with a hefty blast of fresh air, all without any noise! The steady state DC voltage generates 7.5kV DC negative at 400uA, and that's LOTS of ions! Includes 7 wind tubes for max air! Runs on 12-15VDC.

IG7 **Ion Generator Kit** **$64.95**

Tickle-Stick Shocker

The kit has a pulsing 80 volt tickle output and a mischievous blinking LED. And who can resist a blinking light and an unlabeled switch! Great fun for your desk, "Hey, I told you not to touch!" Runs on 3-6 VDC.

TS4 **Tickle Stick Kit** **$9.95**

Speedy Speed Radar Gun

Our famous *Speedy* radar gun teaches you doppler effect the fun way! Digital readout displays in MPH, KPH, or FPS. You supply two coffee cans! Runs on 12VDC or our AC121 supply.

SG7 **Speed Radar Gun Kit** **$74.95**

Tri-Field Meter Kit

"See" electrical, magnetic, and RF fields as a graphical LED display on the front panel! Use it to detect these fields in your house, find RF sources, you name it. Featured on CBS's Ghost Whisperer to detect the presence of

TFM3C **Tri-Field Meter Kit** **$74.95**

Audio Recorder & Player

Record and playback up to 8 minutes of messages from this little board! Built-in condenser mic plus line input, line & speaker outputs. Adjustable sample rate for recording quality. 4-switch operation that can be remote controlled! Runs on 9-12VDC at 500mA.

K8094 **Audio Recorder/Player Kit** **$32.95**

Broadband RF Preamp

Need to "perk-up" your counter or other equipment to read weak signals? This preamp has low noise and yet provides 25dB gain from 1MHz to well over 1GHz. Output can reach 100mW! Runs on 12 volts AC or DC or the included 110VAC PS. Asmb.

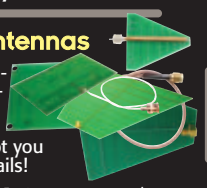
PR2 **Broadband RF Preamp** **$69.95**

Code Practice Oscillator

A great starter kit for young and old! Learn kit building while building a great little CW practice oscillator to learn the code! Built-in key, built in speaker, and audio tone adjust! Runs on a 9V battery.

CPO3 **Code Practice Oscillator Kit** **$17.95**

Precision PC Plane Antennas



Our LPY series PC antennas continue to be the favorite for virtually all RF and wireless applications. From microwave links, wireless mics, to RFID, we've got you covered. Check our site for details!

LPYSeries **Precision PC Plane Antennas** **from $29.95**

Active Receive Antenna

The EASY antenna, regardless of frequency! This active RX antenna performs like a 60' long wire. Works on all bands, and provides over 15dB of gain. Features auto-bypass when turned off. Runs on 9V battery.

AA7C **Active Receive Antenna Kit** **$59.95**

GET THE NUTS AND VOLTS DISCOUNT!

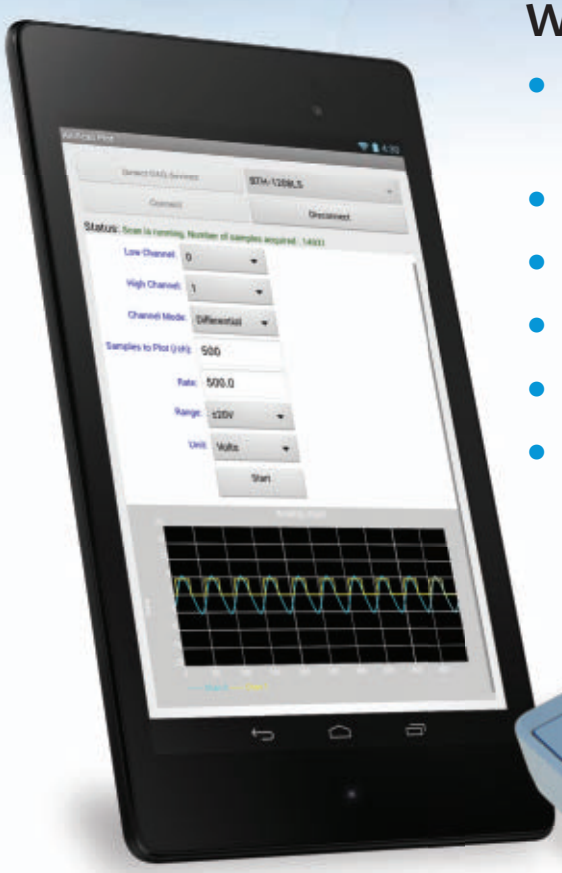
Mention or enter the coupon code ***NVRMZ142*** *and receive 10% off your order!*

800-446-2295
www.ramseykits.com

RAMSEY ELECTRONICS®
590 Fishers Station Drive
Victor, NY 14564
(800) 446-2295
(585) 924-4560

Next Gen Data Logging
Welcome to DAQ 2.0

Use Java™ and Android™ for the Coolest DAQ Apps



Wireless DAQ for Tablets and PCs

- BTH-1208LS multifunction Bluetooth™ DAQ device
- 8 analog input channels
- Rechargeable
- UL for Android Library
- Example programs
- Ready-to-run Demo Apps available at Google Play™

*Collect data
h the
BTH-1208LS DAQ device*

Outstanding Deal – *Only* $199

mccdaq.com/BTH1208LS

MEASUREMENT COMPUTING™

The Value Leader in Data Acquisition

Contact us
1.800.234.4232